SIGGRAPH THINK BEYOND
2020 S2020.SIGGRAPH.ORG

THE TECHNICAL ART OF THE LAST OF US PART II

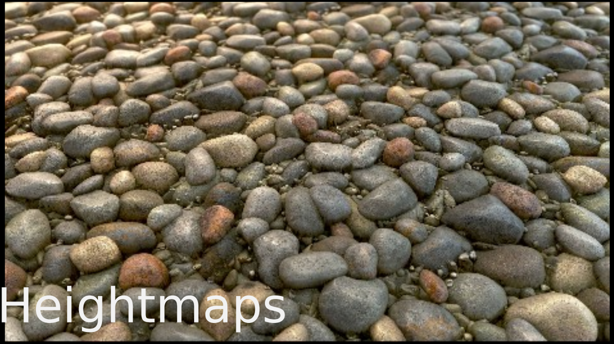
Waylon Brinck — Technical Art Director
Qingzhou “Steven” Tang — Technical Artist

© 2020 SIGGRAPH. ALL RIGHTS RESERVED.

COMICS
MOVIES
TOYS
COSPLAY

Overview

Uber Shader


Heightmaps

Wetness

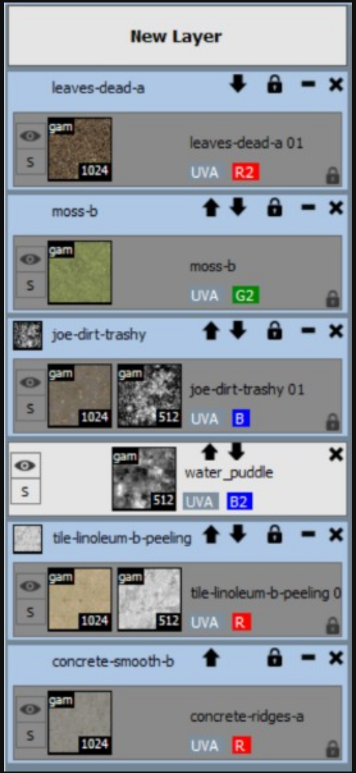
Eyes

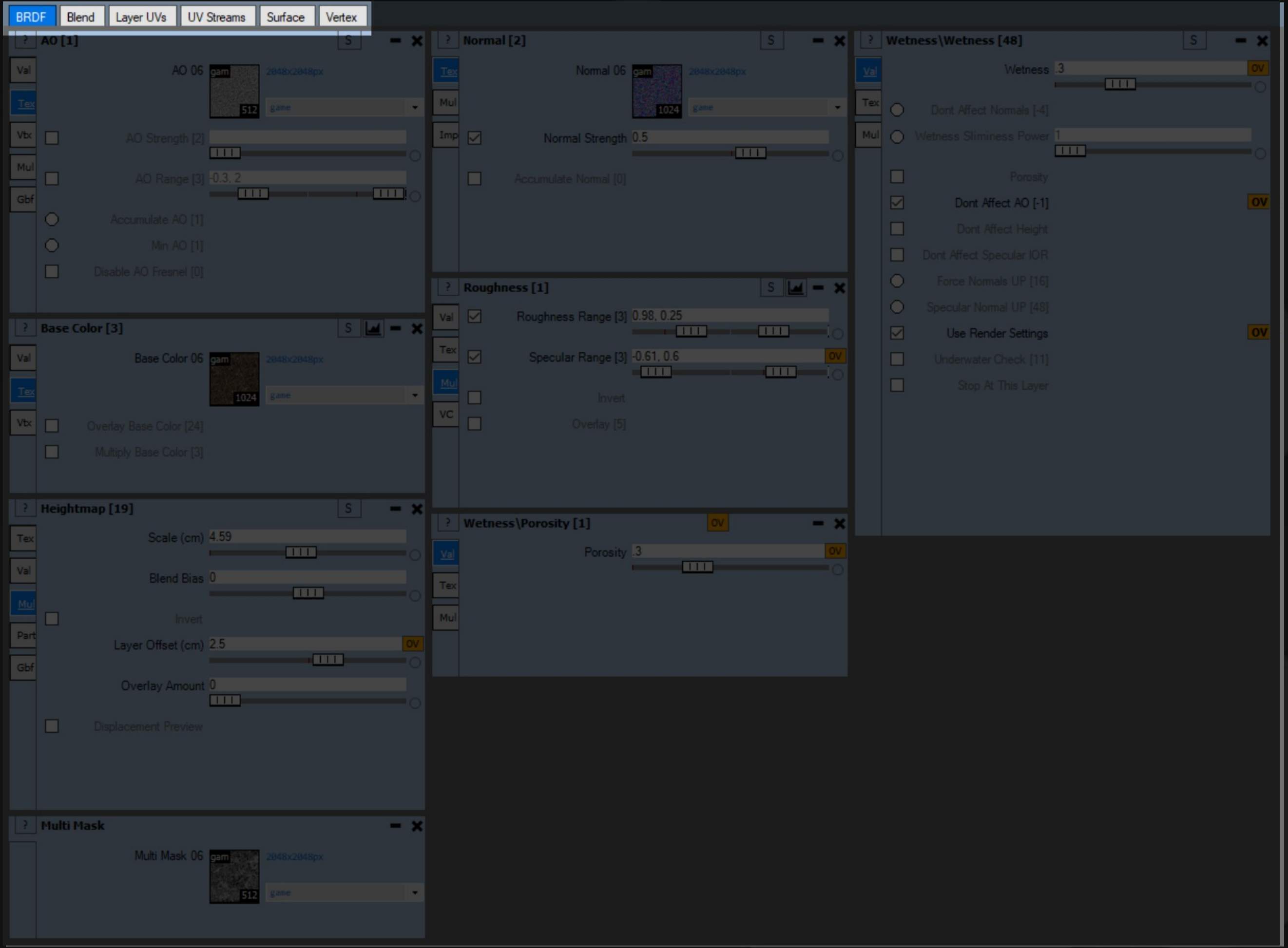
Glass

Snow

THE NAUGHTY DOG UBER SHADER

Leaves


New Layer
leaves-dead-a
gam
1024
S
leaves-dead-a 01
UVA
R2
moss-b
gam
S
moss-b
UVA
G2
joe-dirt-trashy
gam
1024
gam
512
S
joe-dirt-trashy 01
UVA
B
gam
512
S
water_puddle
UVA
B2
tile-linoleum-b-peeling
gam
1024
gam
512
S
tile-linoleum-b-peeling 0
UVA
R
concrete-smooth-b
gam
1024
S
concrete-ridges-a
UVA
R


BRDF
Blend
Layer UVs
UV Streams
Surface
Vertex
?
S
Val
Tex
Vtx
Mul
Gbf
Imp
VC
Part
OV

AO [1]

AO 06 — 512 — 2048x2048px — game

☐ AO Strength [2]

☐ AO Range [3] -0.3, 2

○ Accumulate AO [1]

○ Min AO [1]

☐ Disable AO Fresnel [0]

Base Color [3]

Base Color 06 — 1024 — 2048x2048px — game

☐ Overlay Base Color [24]

☐ Multiply Base Color [3]

Heightmap [19]

Scale (cm) 4.59

Blend Bias 0

☐ Invert

Layer Offset (cm) 2.5 OV

Overlay Amount 0

☐ Displacement Preview

Multi Mask

Multi Mask 06 — 512 — 2048x2048px — game

Normal [2]

Normal 06 — 1024 — 2048x2048px — game

☑ Normal Strength 0.5

☐ Accumulate Normal [0]

Roughness [1]

☑ Roughness Range [3] 0.98, 0.25

☑ Specular Range [3] -0.61, 0.6 OV

☐ Invert

☐ Overlay [5]

Wetness\Porosity [1] OV

Porosity .3 OV

Wetness\Wetness [48]

Wetness .3 OV

○ Dont Affect Normals [-4]

○ Wetness Sliminess Power 1

☐ Porosity

☑ Dont Affect AO [-1] OV

☐ Dont Affect Height

☐ Dont Affect Specular IOR

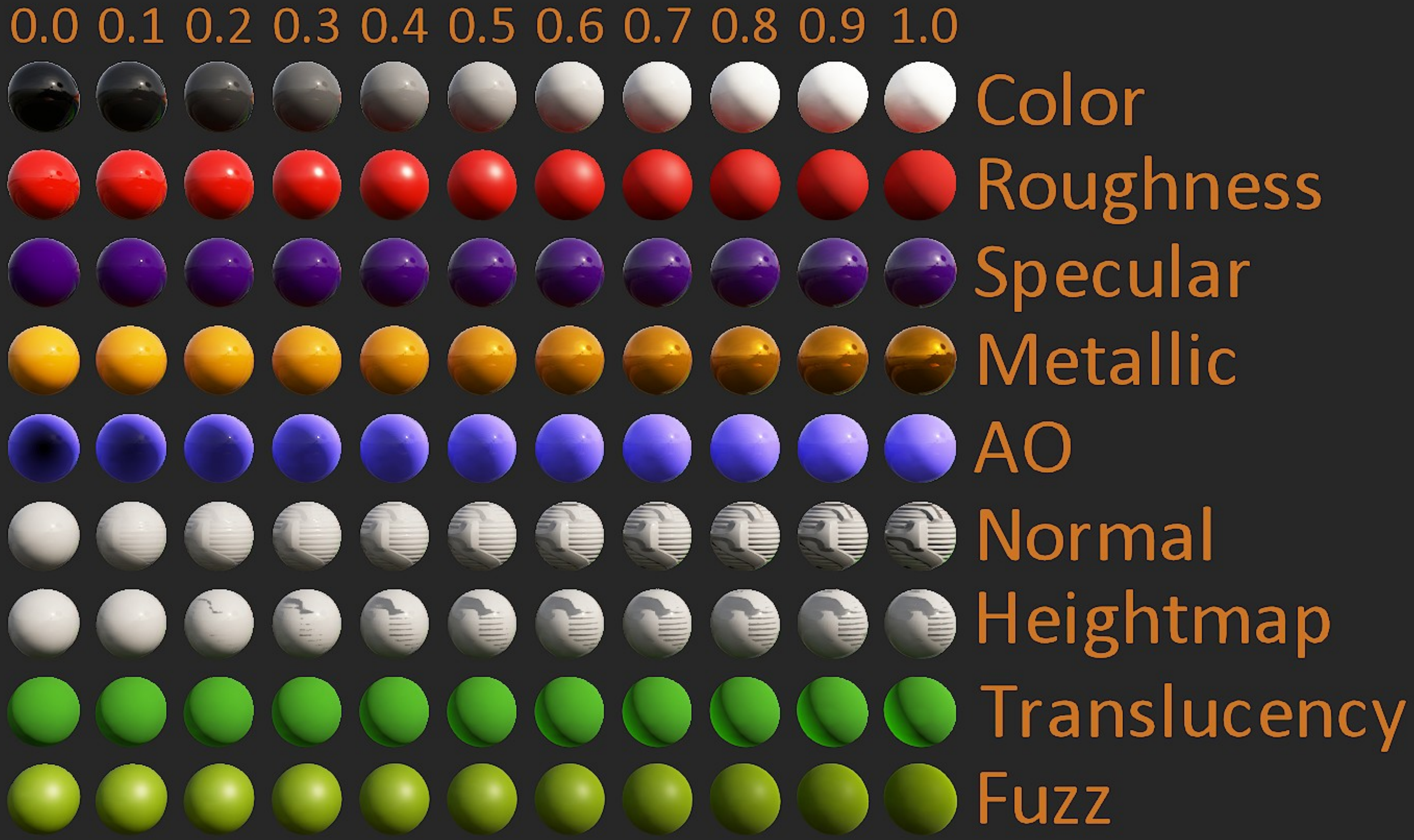
○ Force Normals UP [16]

○ Specular Normal UP [48]

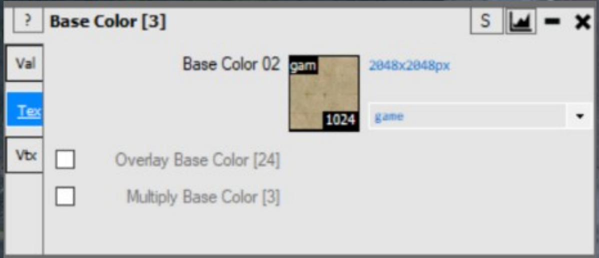
☑ Use Render Settings OV

☐ Underwater Check [11]

☐ Stop At This Layer


0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Color
Roughness
Specular
Metallic
AO
Normal
Heightmap
Translucency
Fuzz

BRD
F

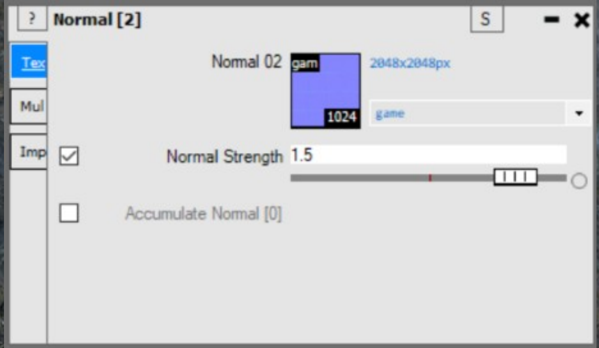

Base Color [3]

?
S
Val
Tex
Vtx
gam
1024
game

Base Color 02 2048x2048px

☐ Overlay Base Color [24]

☐ Multiply Base Color [3]

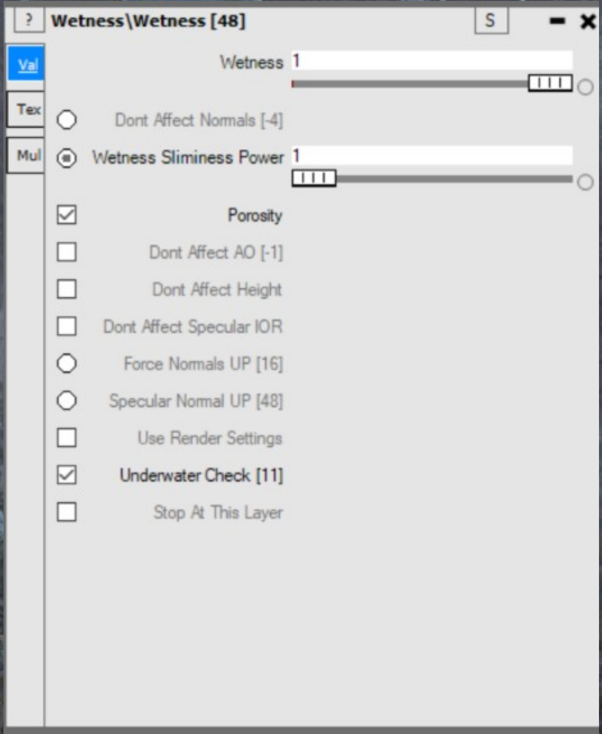

Normal [2]

?
S
Tex
Mul
Imp
gam
1024
game

Normal 02 2048x2048px

☑ Normal Strength 1.5

☐ Accumulate Normal [0]


Wetness\Wetness [48]

?
S
Val
Tex
Mul

Wetness 1

○ Dont Affect Normals [-4]

● Wetness Sliminess Power 1

☑ Porosity

☐ Dont Affect AO [-1]

☐ Dont Affect Height

☐ Dont Affect Specular IOR

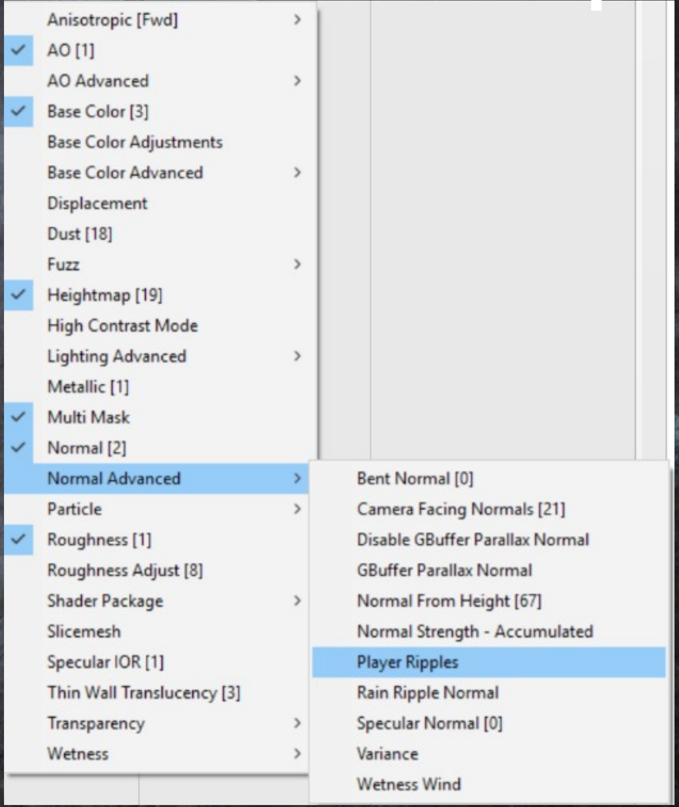
○ Force Normals UP [16]

○ Specular Normal UP [48]

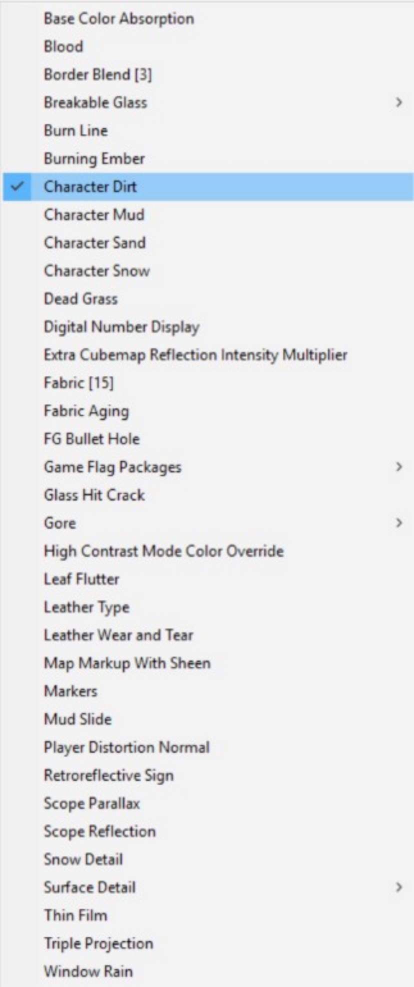
☐ Use Render Settings

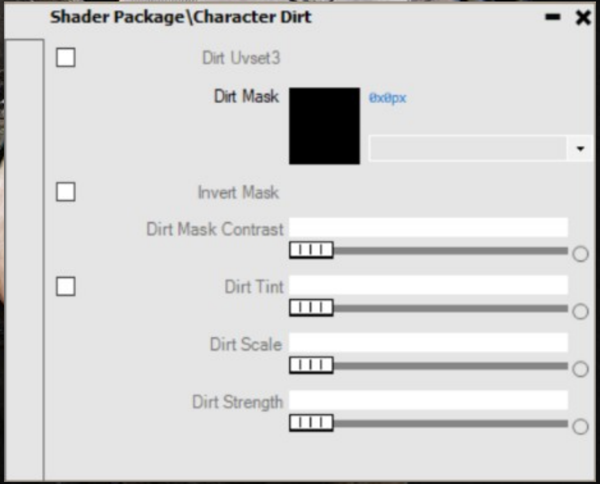
☑ Underwater Check [11]

☐ Stop At This Layer

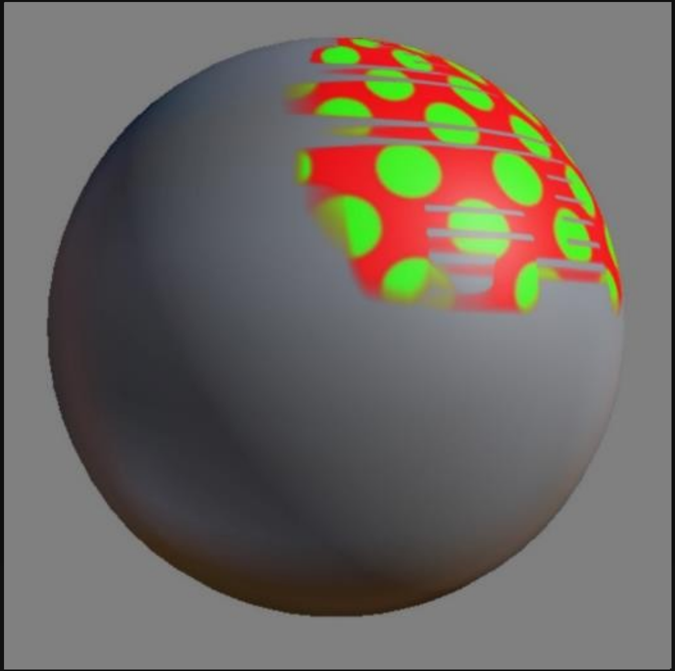

Anisotropic [Fwd]
✓ AO [1]
AO Advanced
✓ Base Color [3]
Base Color Adjustments
Base Color Advanced
Displacement
Dust [18]
Fuzz
✓ Heightmap [19]
High Contrast Mode
Lighting Advanced
Metallic [1]
✓ Multi Mask
✓ Normal [2]
Normal Advanced
Particle
✓ Roughness [1]
Roughness Adjust [8]
Shader Package
Slicemesh
Specular IOR [1]
Thin Wall Translucency [3]
Transparency
Wetness
Bent Normal [0]
Camera Facing Normals [21]
Disable GBuffer Parallax Normal
GBuffer Parallax Normal
Normal From Height [67]
Normal Strength - Accumulated
Player Ripples
Rain Ripple Normal
Specular Normal [0]
Variance
Wetness Wind

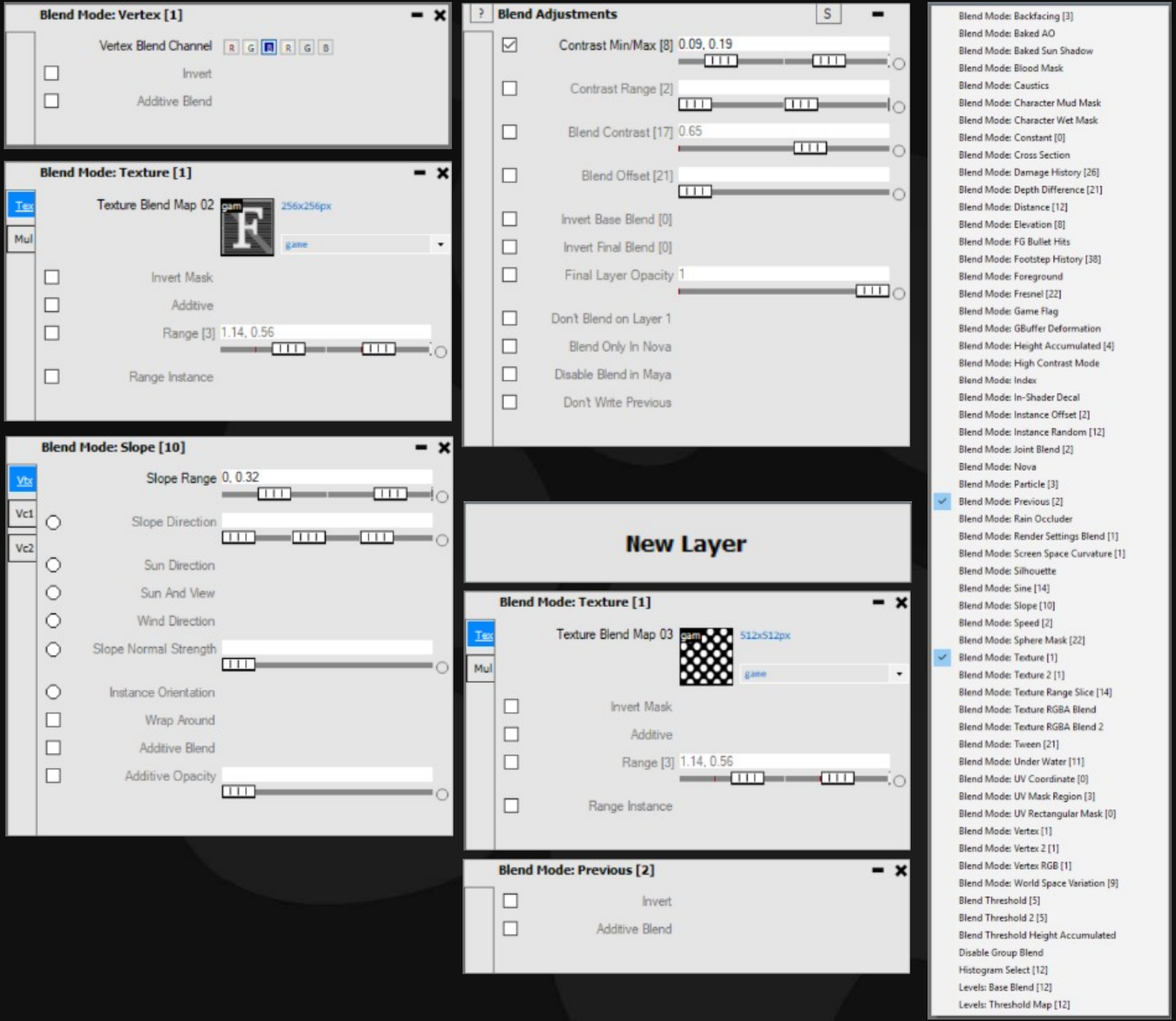
BRDF – Shader Packages


Base Color Absorption
Blood
Border Blend [3]
Breakable Glass ›
Burn Line
Burning Ember
✓ Character Dirt
Character Mud
Character Sand
Character Snow
Dead Grass
Digital Number Display
Extra Cubemap Reflection Intensity Multiplier
Fabric [15]
Fabric Aging
FG Bullet Hole
Game Flag Packages ›
Glass Hit Crack
Gore ›
High Contrast Mode Color Override
Leaf Flutter
Leather Type
Leather Wear and Tear
Map Markup With Sheen
Markers
Mud Slide
Player Distortion Normal
Retroreflective Sign
Scope Parallax
Scope Reflection
Snow Detail
Surface Detail ›
Thin Film
Triple Projection
Window Rain


Shader Package\Character Dirt
☐ Dirt Uvset3
Dirt Mask
0x0px
☐ Invert Mask
Dirt Mask Contrast
☐ Dirt Tint
Dirt Scale
Dirt Strength

Blends





Blend Mode: Vertex [1]

Vertex Blend Channel: R G B R G B

☐ Invert

☐ Additive Blend

Blend Mode: Texture [1]

Texture Blend Map 02 — 256x256px — game

☐ Invert Mask

☐ Additive

☐ Range [3] 1.14, 0.56

☐ Range Instance

Blend Mode: Slope [10]

Slope Range 0, 0.32

○ Slope Direction

○ Sun Direction

○ Sun And View

○ Wind Direction

○ Slope Normal Strength

○ Instance Orientation

☐ Wrap Around

☐ Additive Blend

☐ Additive Opacity

Blend Adjustments

☑ Contrast Min/Max [8] 0.09, 0.19

☐ Contrast Range [2]

☐ Blend Contrast [17] 0.65

☐ Blend Offset [21]

☐ Invert Base Blend [0]

☐ Invert Final Blend [0]

☐ Final Layer Opacity 1

☐ Don't Blend on Layer 1

☐ Blend Only In Nova

☐ Disable Blend in Maya

☐ Don't Write Previous

New Layer

Blend Mode: Texture [1]

Texture Blend Map 03 — 512x512px — game

☐ Invert Mask

☐ Additive

☐ Range [3] 1.14, 0.56

☐ Range Instance

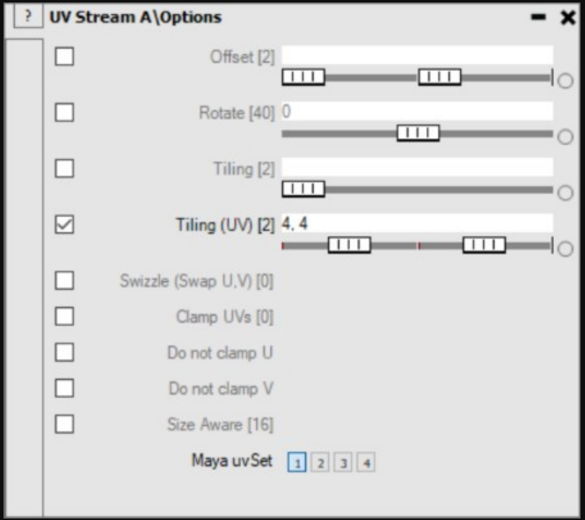
Blend Mode: Previous [2]

☐ Invert

☐ Additive Blend

Tex
Mul
Vtx
Vc1
Vc2
gam
F
?
S
Blend Mode: Backfacing [3]
Blend Mode: Baked AO
Blend Mode: Baked Sun Shadow
Blend Mode: Blood Mask
Blend Mode: Caustics
Blend Mode: Character Mud Mask
Blend Mode: Character Wet Mask
Blend Mode: Constant [0]
Blend Mode: Cross Section
Blend Mode: Damage History [26]
Blend Mode: Depth Difference [21]
Blend Mode: Distance [12]
Blend Mode: Elevation [8]
Blend Mode: FG Bullet Hits
Blend Mode: Footstep History [38]
Blend Mode: Foreground
Blend Mode: Fresnel [22]
Blend Mode: Game Flag
Blend Mode: GBuffer Deformation
Blend Mode: Height Accumulated [4]
Blend Mode: High Contrast Mode
Blend Mode: Index
Blend Mode: In-Shader Decal
Blend Mode: Instance Offset [2]
Blend Mode: Instance Random [12]
Blend Mode: Joint Blend [2]
Blend Mode: Nova
Blend Mode: Particle [3]
✓ Blend Mode: Previous [2]
Blend Mode: Rain Occluder
Blend Mode: Render Settings Blend [1]
Blend Mode: Screen Space Curvature [1]
Blend Mode: Silhouette
Blend Mode: Sine [14]
Blend Mode: Slope [10]
Blend Mode: Speed [2]
Blend Mode: Sphere Mask [22]
✓ Blend Mode: Texture [1]
Blend Mode: Texture 2 [1]
Blend Mode: Texture Range Slice [14]
Blend Mode: Texture RGBA Blend
Blend Mode: Texture RGBA Blend 2
Blend Mode: Tween [21]
Blend Mode: Under Water [11]
Blend Mode: UV Coordinate [0]
Blend Mode: UV Mask Region [3]
Blend Mode: UV Rectangular Mask [0]
Blend Mode: Vertex [1]
Blend Mode: Vertex 2 [1]
Blend Mode: Vertex RGB [1]
Blend Mode: World Space Variation [9]
Blend Threshold [5]
Blend Threshold 2 [5]
Blend Threshold Height Accumulated
Disable Group Blend
Histogram Select [12]
Levels: Base Blend [12]
Levels: Threshold Map [12]

UV Streams



UV Stream A\Options

☐ Offset [2]

☐ Rotate [40] 0

☐ Tiling [2]

☑ Tiling (UV) [2] 4, 4

☐ Swizzle (Swap U,V) [0]

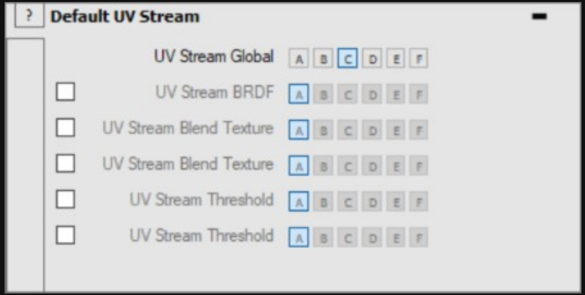
☐ Clamp UVs [0]

☐ Do not clamp U

☐ Do not clamp V

☐ Size Aware [16]

Maya uvSet 1 2 3 4



Default UV Stream

UV Stream Global A B C D E F

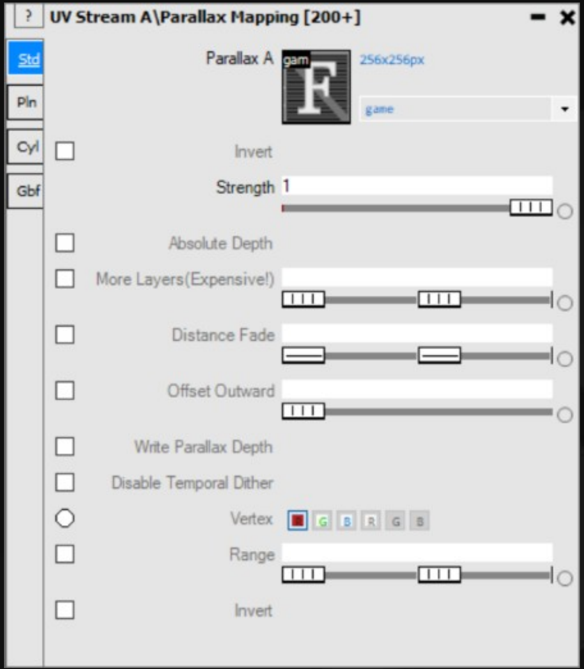
☐ UV Stream BRDF A B C D E F

☐ UV Stream Blend Texture A B C D E F

☐ UV Stream Blend Texture A B C D E F

☐ UV Stream Threshold A B C D E F

☐ UV Stream Threshold A B C D E F



UV Stream A\Parallax Mapping [200+]

Std
Pln
Cyl
Gbf

Parallax A gam 256x256px game

☐ Invert

Strength 1

☐ Absolute Depth

☐ More Layers(Expensive!)

☐ Distance Fade

☐ Offset Outward

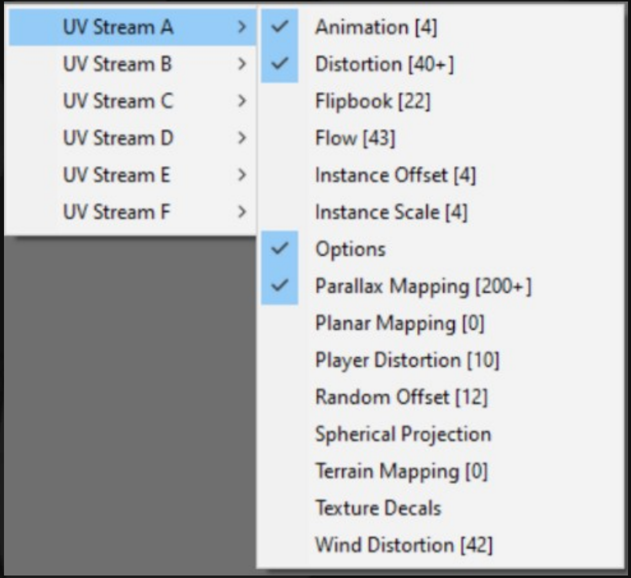
☐ Write Parallax Depth

☐ Disable Temporal Dither

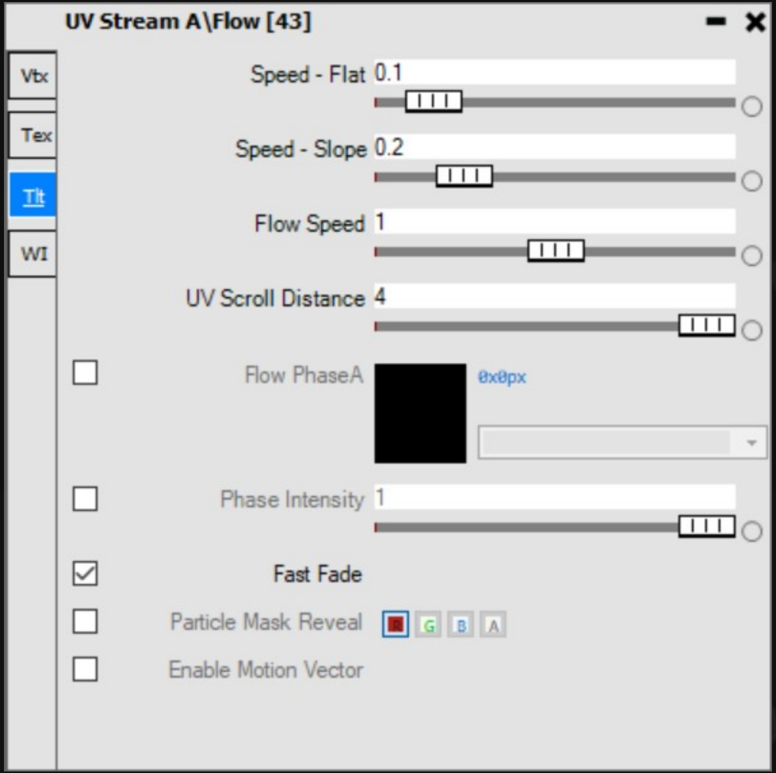
○ Vertex R G B R G B

☐ Range

☐ Invert



UV Stream A
UV Stream B
UV Stream C
UV Stream D
UV Stream E
UV Stream F
✓ Animation [4]
✓ Distortion [40+]
Flipbook [22]
Flow [43]
Instance Offset [4]
Instance Scale [4]
✓ Options
✓ Parallax Mapping [200+]
Planar Mapping [0]
Player Distortion [10]
Random Offset [12]
Spherical Projection
Terrain Mapping [0]
Texture Decals
Wind Distortion [42]



UV Stream A\Flow [43]

Vtx
Tex
Tlt
WI
0x0px

Speed - Flat 0.1

Speed - Slope 0.2

Flow Speed 1

UV Scroll Distance 4

☐ Flow PhaseA

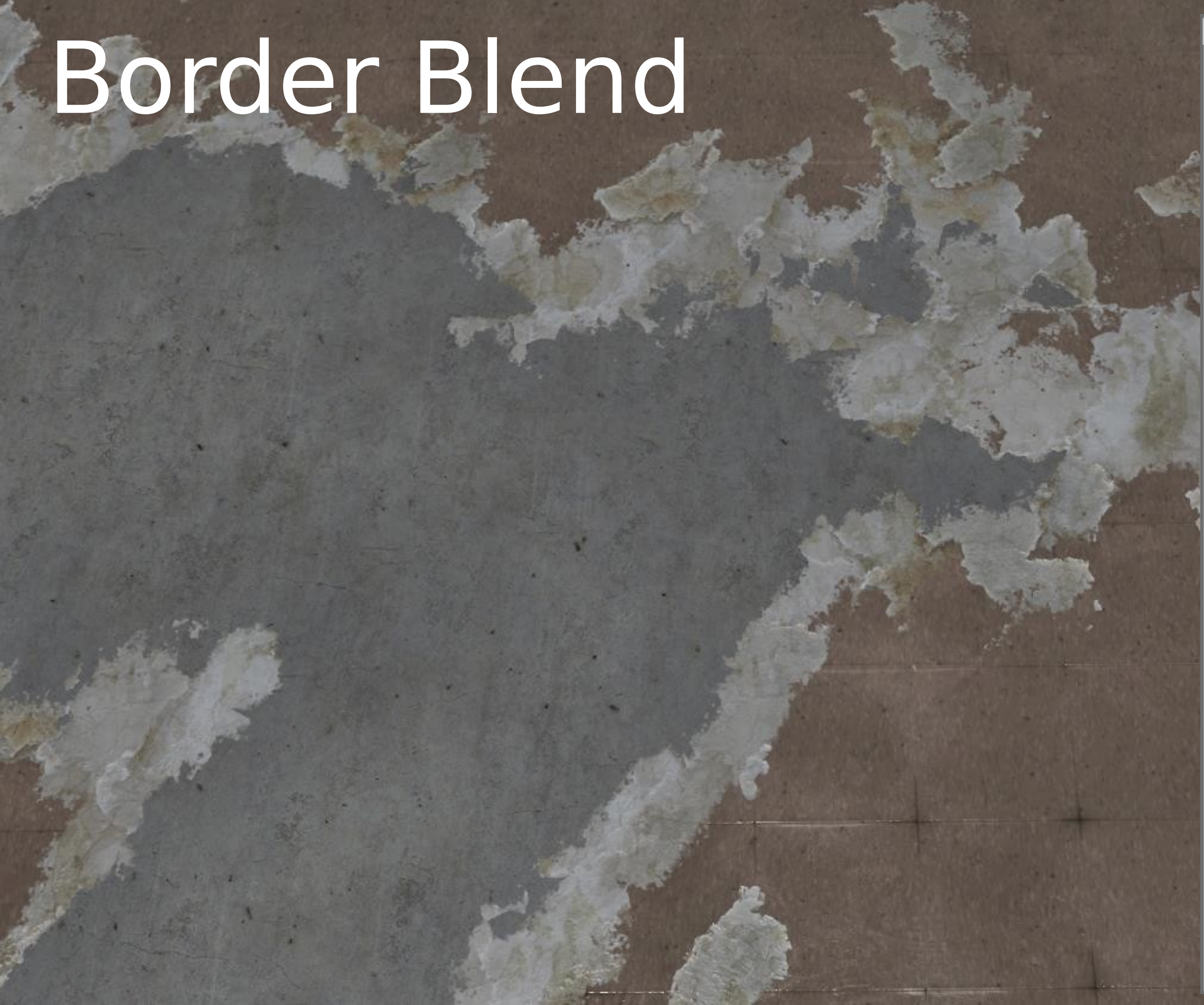
☐ Phase Intensity 1

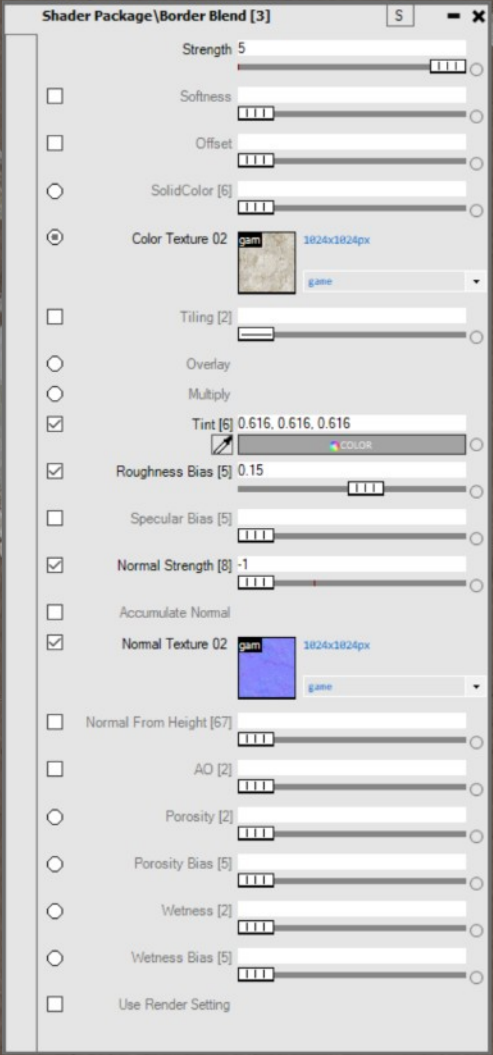
☑ Fast Fade

☐ Particle Mask Reveal R G B A

☐ Enable Motion Vector

Border Blend

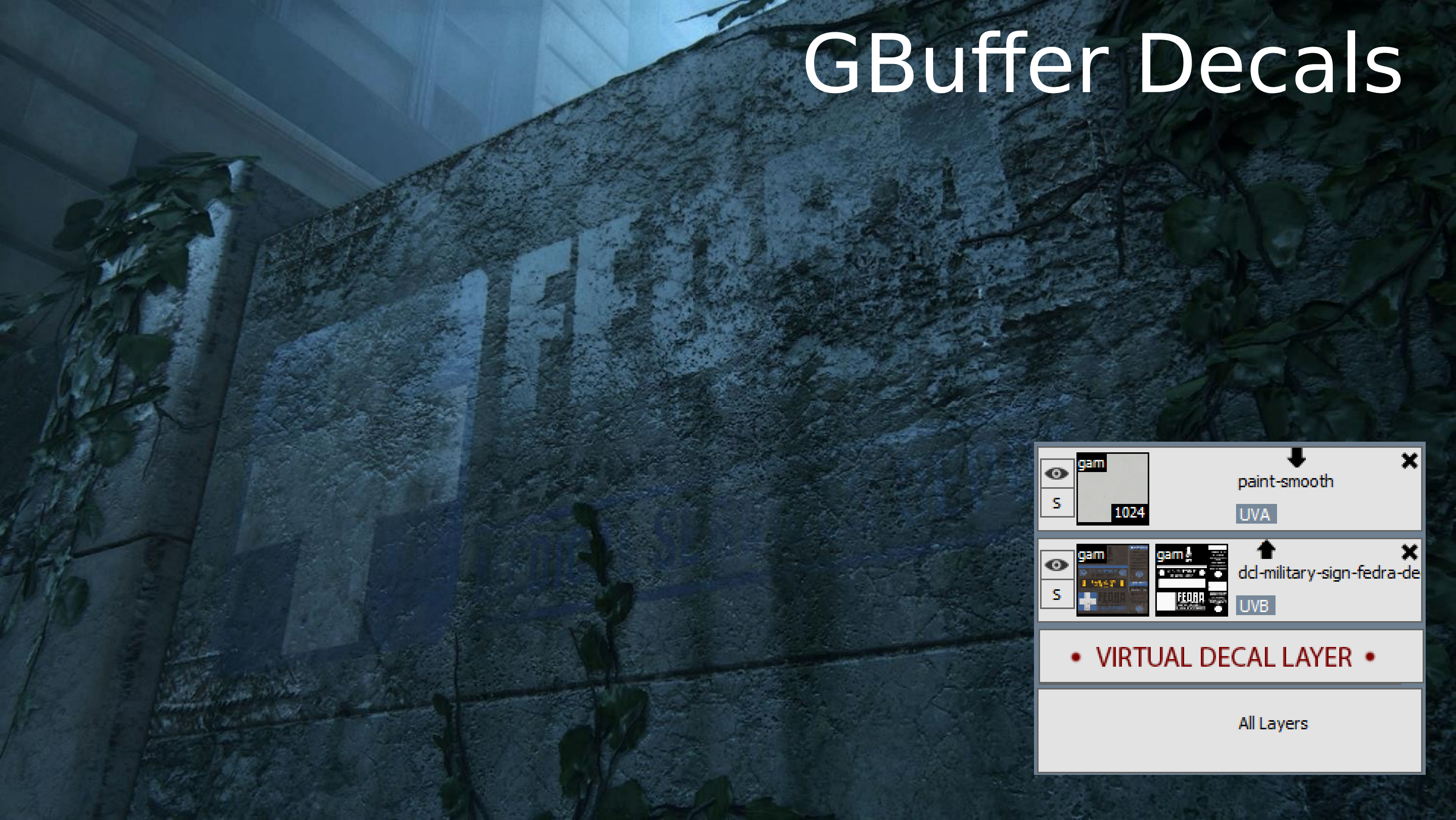




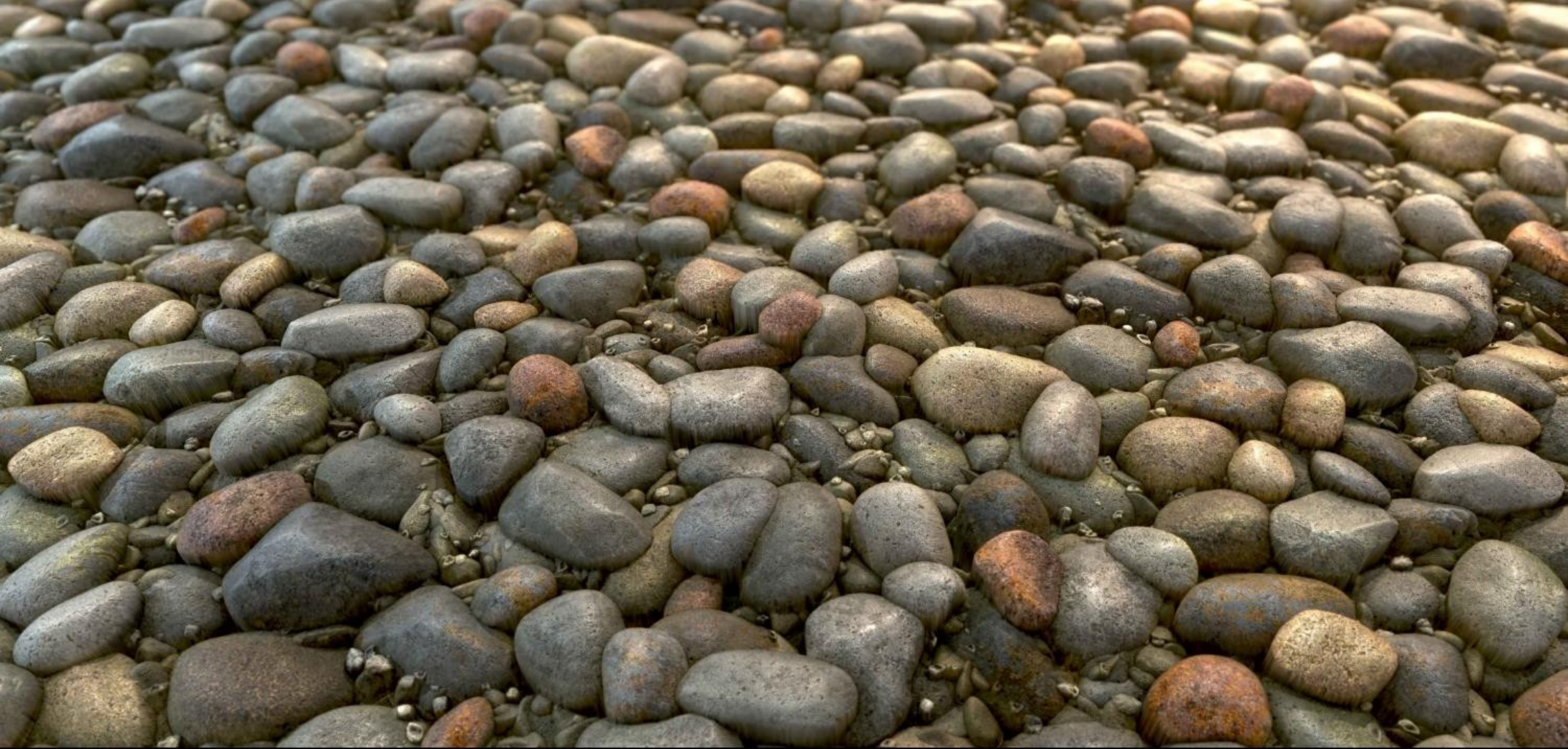
Shader Package\Border Blend [3]

Strength 5
☐ Softness
☐ Offset
○ SolidColor [6]
● Color Texture 02 1024x1024px game
☐ Tiling [2]
○ Overlay
○ Multiply
☑ Tint [6] 0.616, 0.616, 0.616
☑ Roughness Bias [5] 0.15
☐ Specular Bias [5]
☑ Normal Strength [8] -1
☐ Accumulate Normal
☑ Normal Texture 02 1024x1024px game
☐ Normal From Height [67]
☐ AO [2]
○ Porosity [2]
○ Porosity Bias [5]
○ Wetness [2]
○ Wetness Bias [5]
☐ Use Render Setting

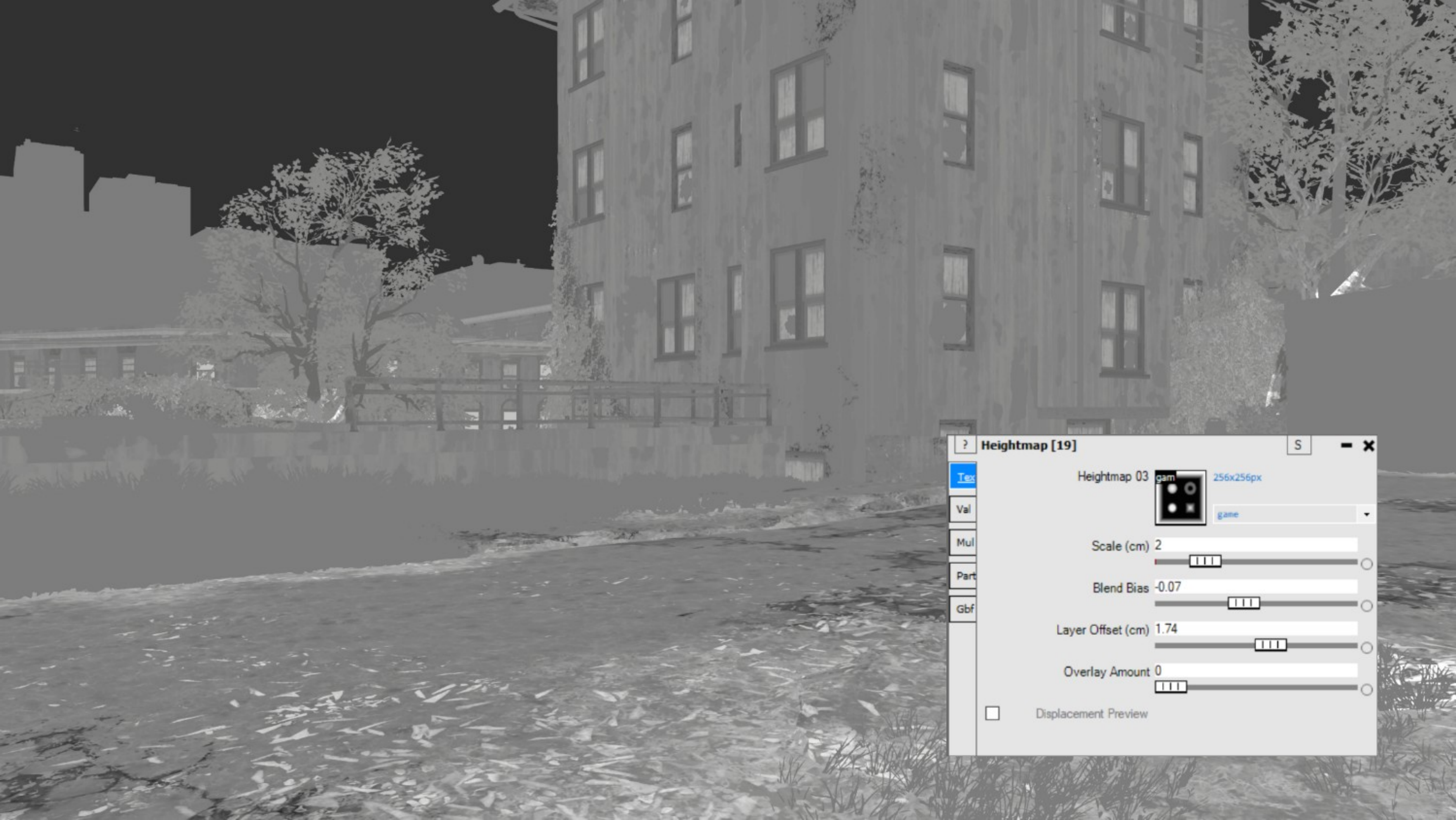
GBuffer Decals


S
gam
1024
paint-smooth
UVA
S
gam
gam
dcl-military-sign-fedra-de
UVB
• VIRTUAL DECAL LAYER •
All Layers

THE NAUGHTY DOG UBER SHADER ✓



HEIGHTMAPS



?
S
Tex
Val
Mul
Part
Gbf
gam
256x256px

Heightmap [19]

Heightmap 03: game

Scale (cm): 2

Blend Bias: -0.07

Layer Offset (cm): 1.74

Overlay Amount: 0

☐ Displacement Preview

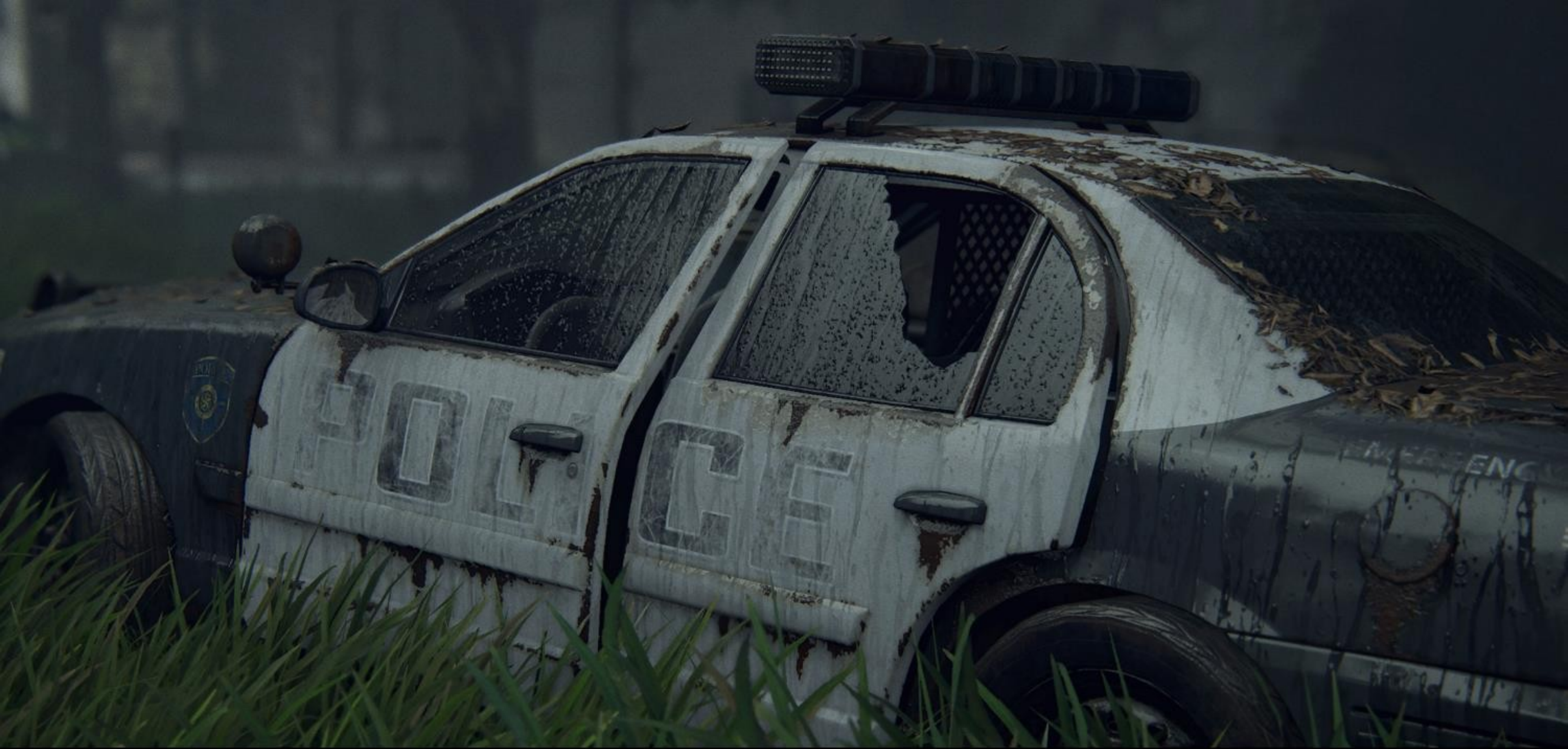
Screen-Space

W.L.F

Derivative Normals

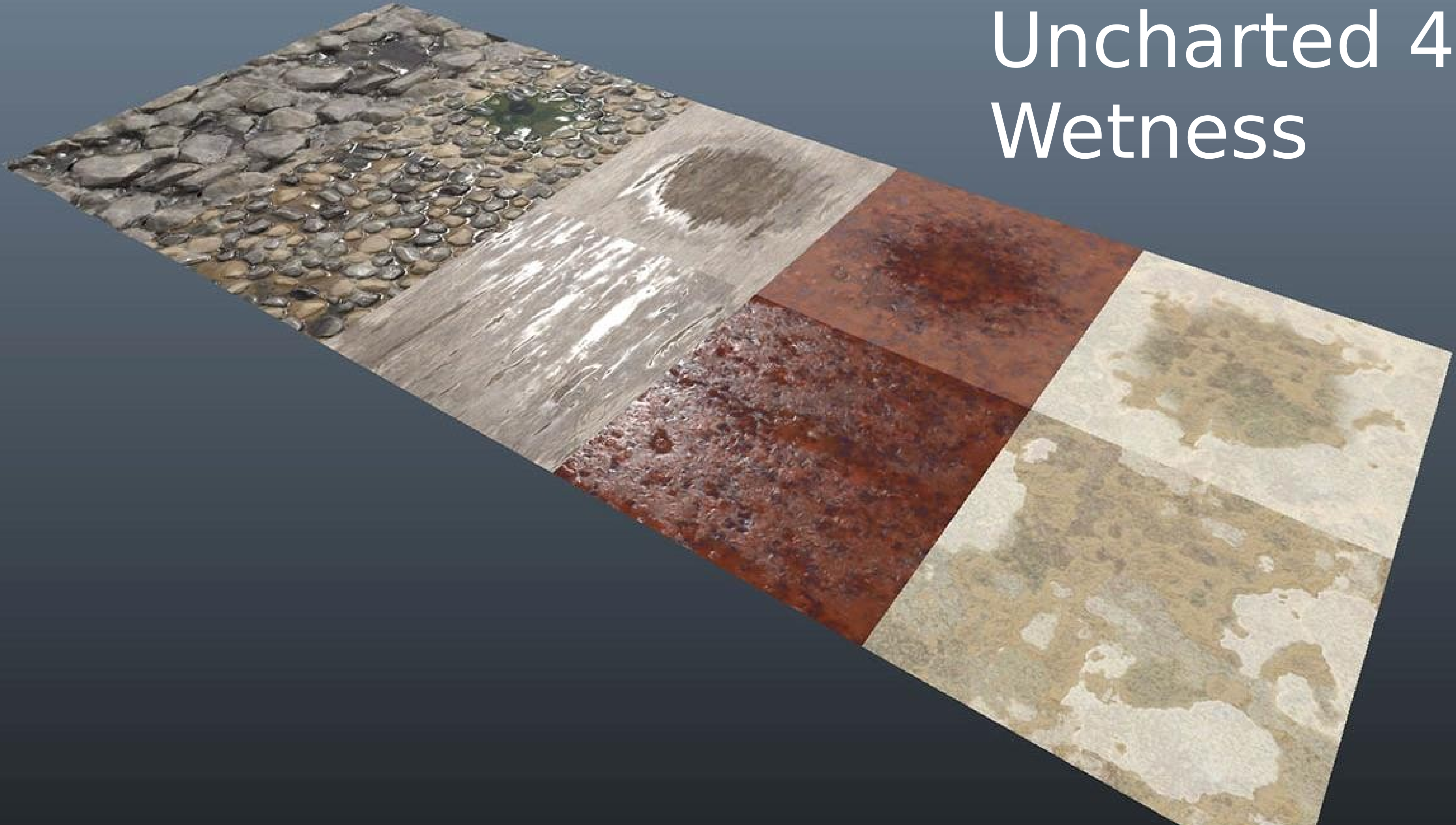
```
float3 GetNormalFromHeightWSFaceted(float3 vertexNormalWS, float heightInMeters)
{
  float3 positionWS = g_positionWS + heightInMeters * vertexNormalWS;
  float3 tangentWS = ddx(positionWS);
  float3 bitangentWS = ddy(positionWS);

  float3 normalWS = normalize(cross(bitangentWS, tangentWS));

  // Result contains polygonal faceting
  return normalWS;
}

float3 GetNormalFromHeightWS(float3 vertexNormalWS, float heightInMeters)
{
  float3 normalWS = GetNormalFromHeightWSFaceted(vertexNormalWS, heightInMeters);

  // Replace the faceted normal with the smooth vertex normal
  normalWS = normalWS - GetNormalFromHeightWSFaceted(vertexNormalWS, 0) + vertexNormalWS;
  return normalWS;
}

// Calculate and apply the normal in the blend region
float3 ApplyDerivativeNormal(float3 vertexNormalWS, float heightInMeters, float blendMask)
{
  float3 heightNormalWS = GetNormalFromHeightWS(vertexNormalWS, heightInMeters);
  float3 finalNormalWS = normalize(normalWS + heightNormalWS * blendMask);
}
```

HEIGHTMAPS ✓

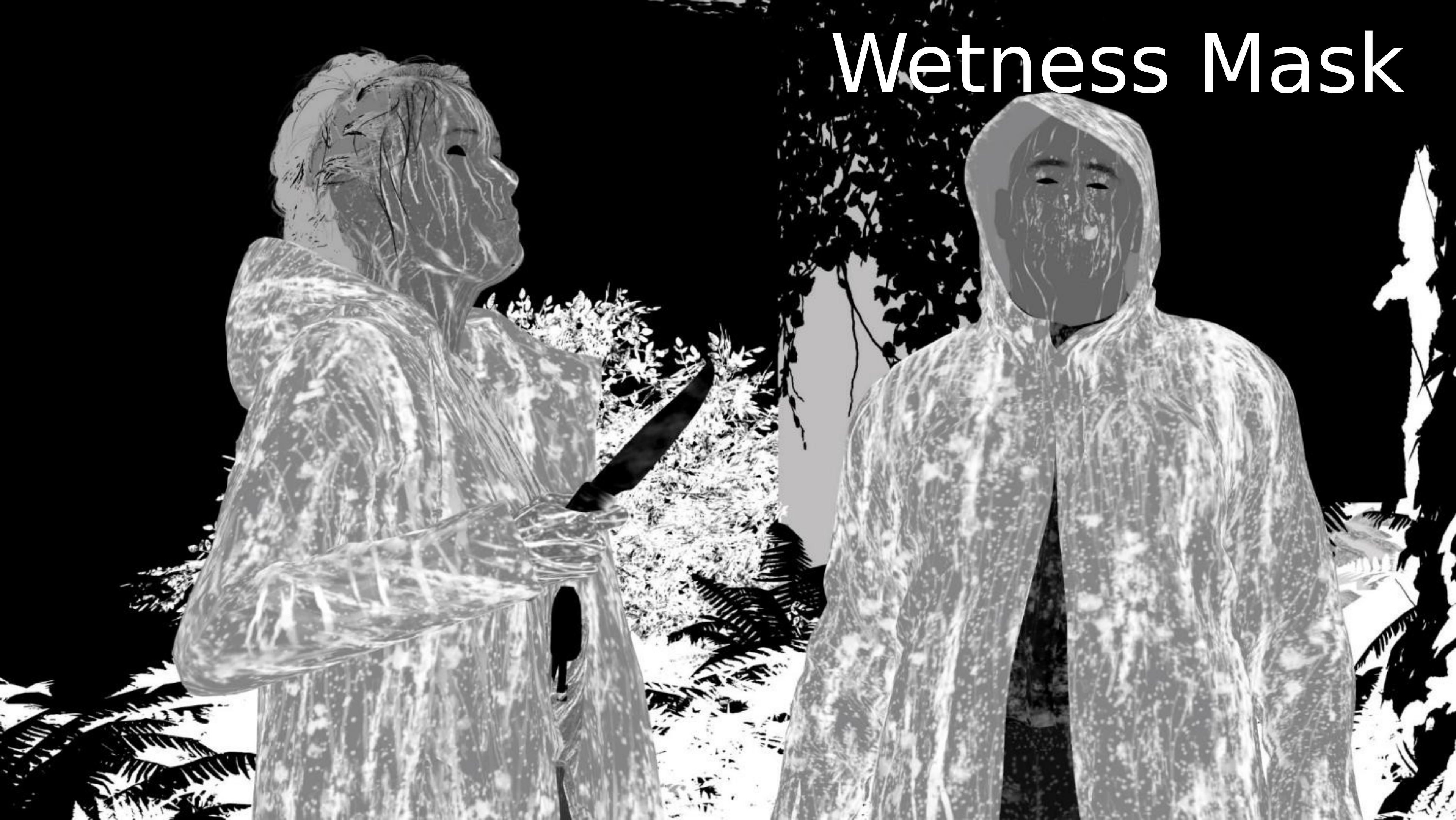

POLICE
EMERGENC

WETNESS

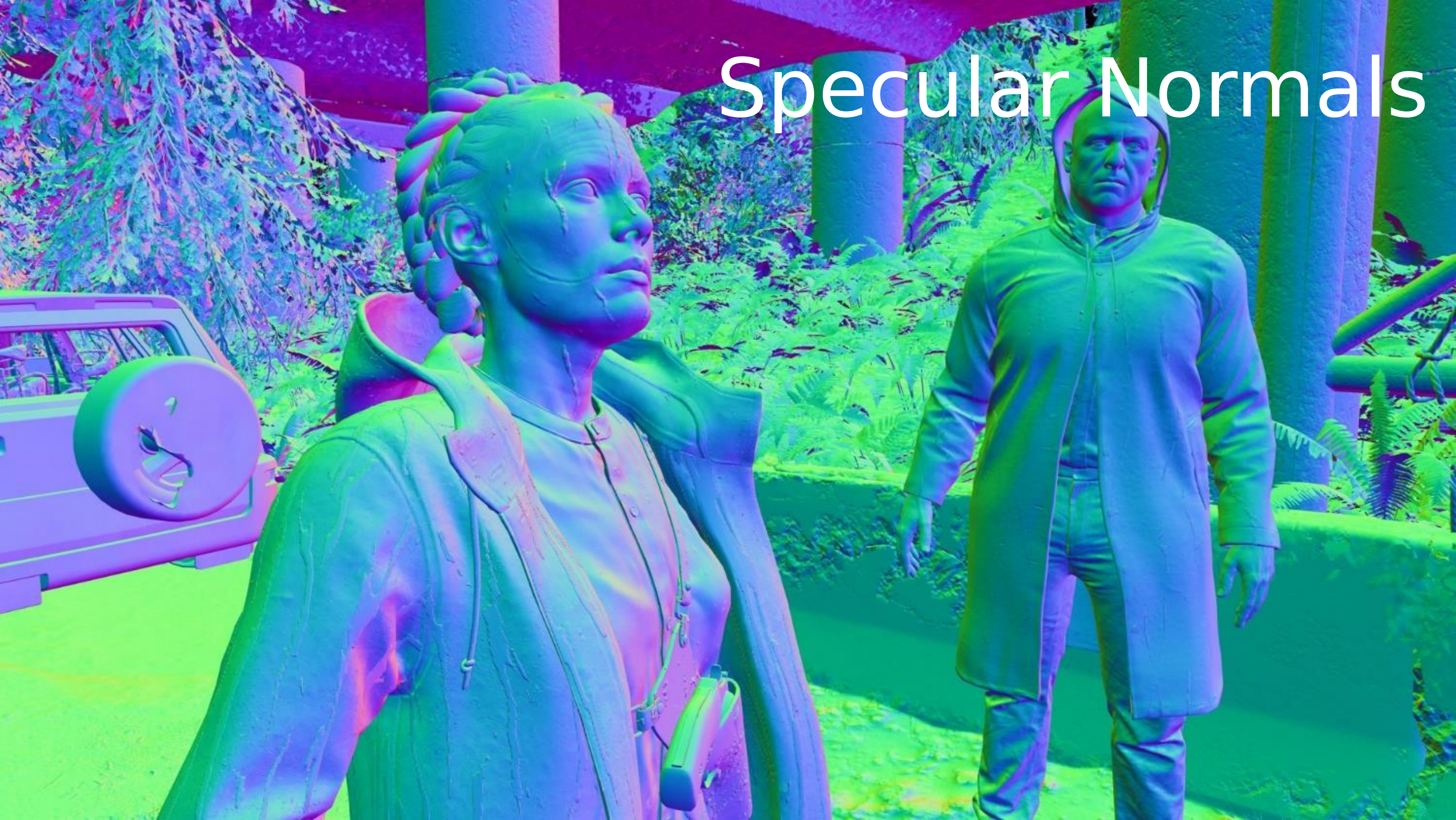
Uncharted 4 Wetness



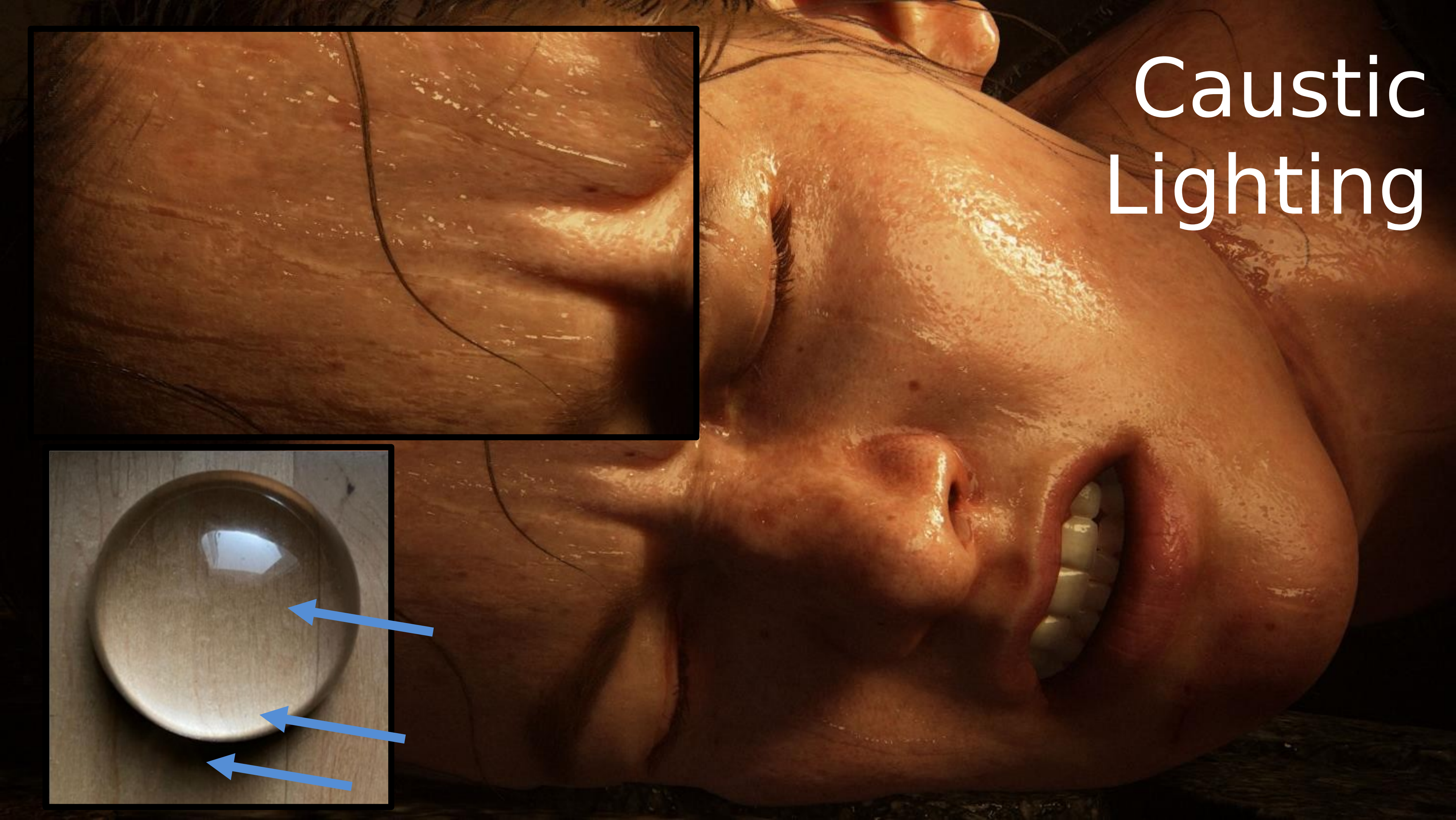
Wetness Mask



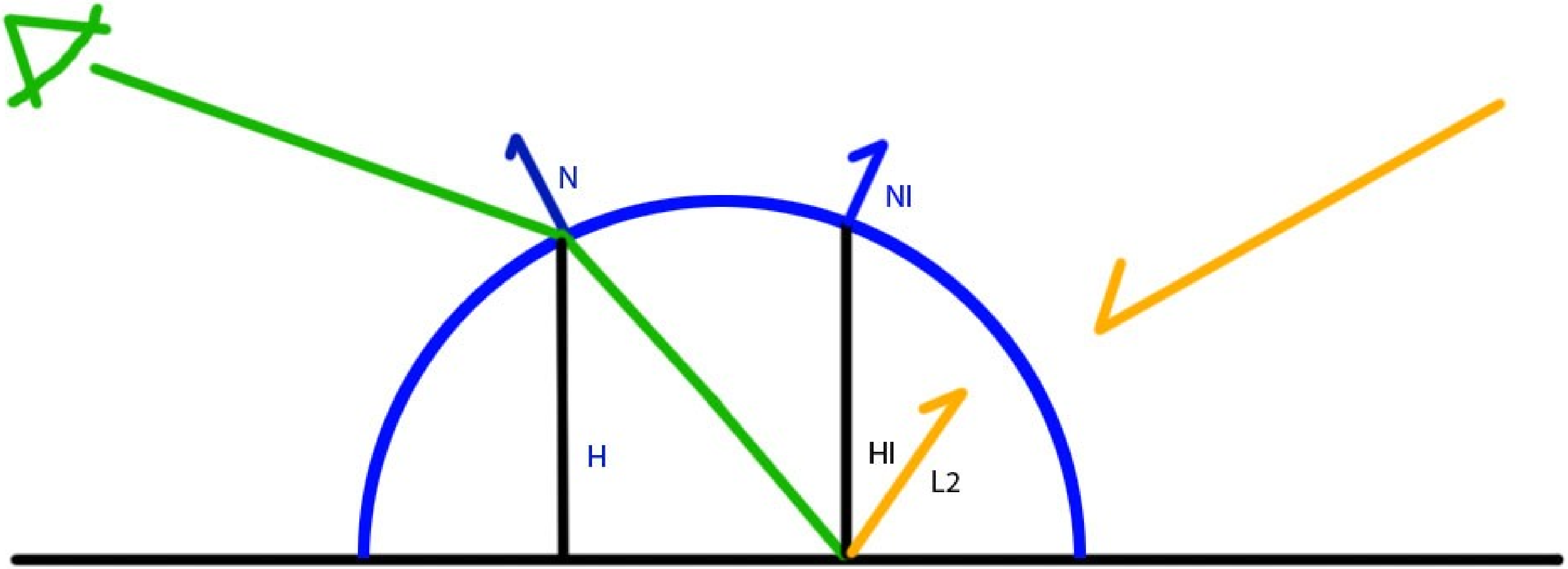
Specular Normals



Caustic Lighting



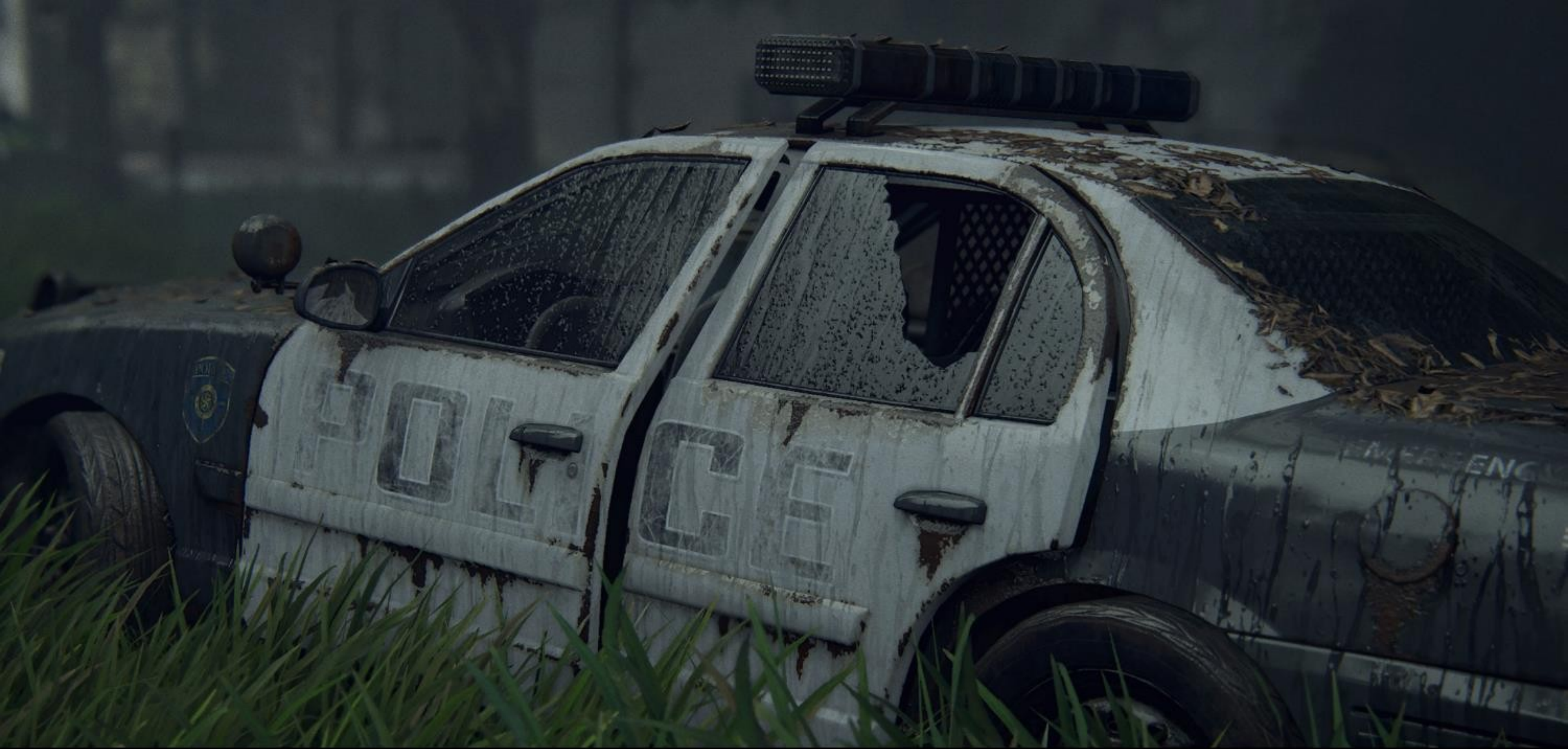
Mathematical Model of the



... Given only the normal and heigh

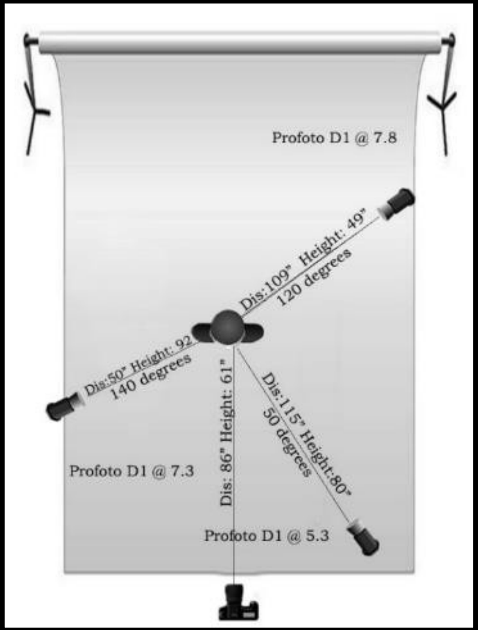
```
// Calculate caustics from a layer of wetness inside the BRDF
float DiffuseSpecularCaustics(const Setup setup, float3 lightDirWS)
{
float cosThetaL = saturate(dot(setup.normalWS, lightDirWS));
float cosThetaLSpecUnclamped = dot(setup.specularNormalWS, lightDirWS);
float cosThetaLSpec = saturate(cosThetaLSpecUnclamped);

// Refractive caustics term (brightening and darkening), artistically tuned against reference
float caustic = 0.8 + cosThetaL * 0.2 + Pow8(1.0 - cosThetaLSpec) * 7.0;

// Simulates refractive shadowing, artistically tuned against reference
float fresnelOcclusion = saturate(1.05 - Pow5(1.0 - cosThetaLSpecUnclamped)) + 0.05;
caustic *= fresnelOcclusion;

return caustic;
}
```

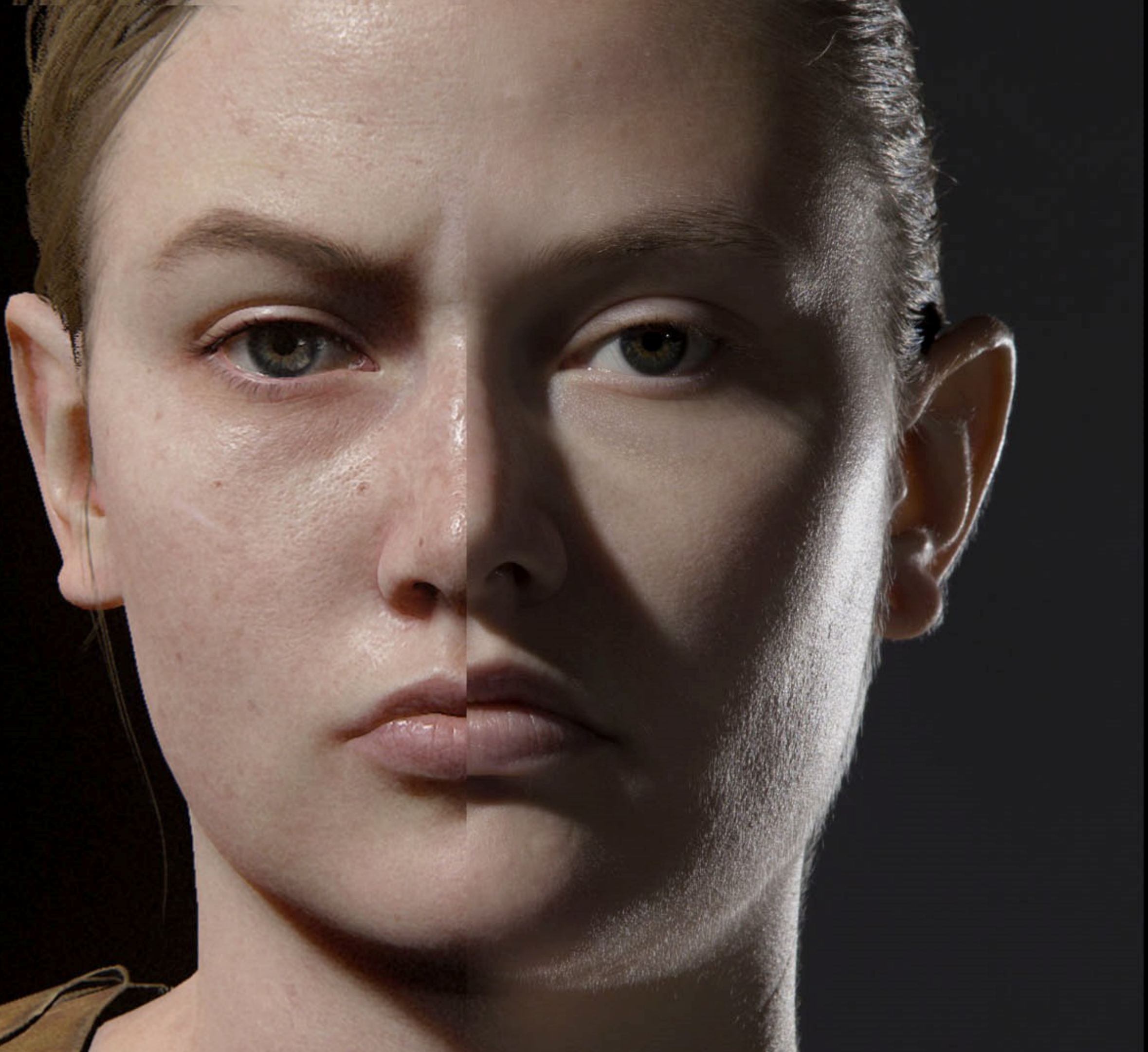
```
// GIANT HACK !!!!!!!!!!!!!!!!!!!!!!!!!!!
float GiantHackCaustics(const Setup setup, float3 lightDirWS)
{
float cosThetaL = saturate(dot(setup.normalWS, lightDirWS));
float cosThetaLSpecUnclamped = dot(setup.specularNormalWS, lightDirWS);
float cosThetaLSpec = saturate(cosThetaLSpecUnclamped);

// Hacky caustics term (brightening and darkening), artistically hacked against hack
float caustic = 0.8 + cosThetaL * 0.2 + Pow8(1.0 - cosThetaLSpec) * 7.0;

// Hacky refractive shadowing, artistically hack hack hack
float fresnelOcclusion = saturate(1.05 - Pow5(1.0 - cosThetaLSpecUnclamped)) + 0.05;
caustic *= fresnelOcclusion;

// Get a programmer to help!
return caustic;
}
```

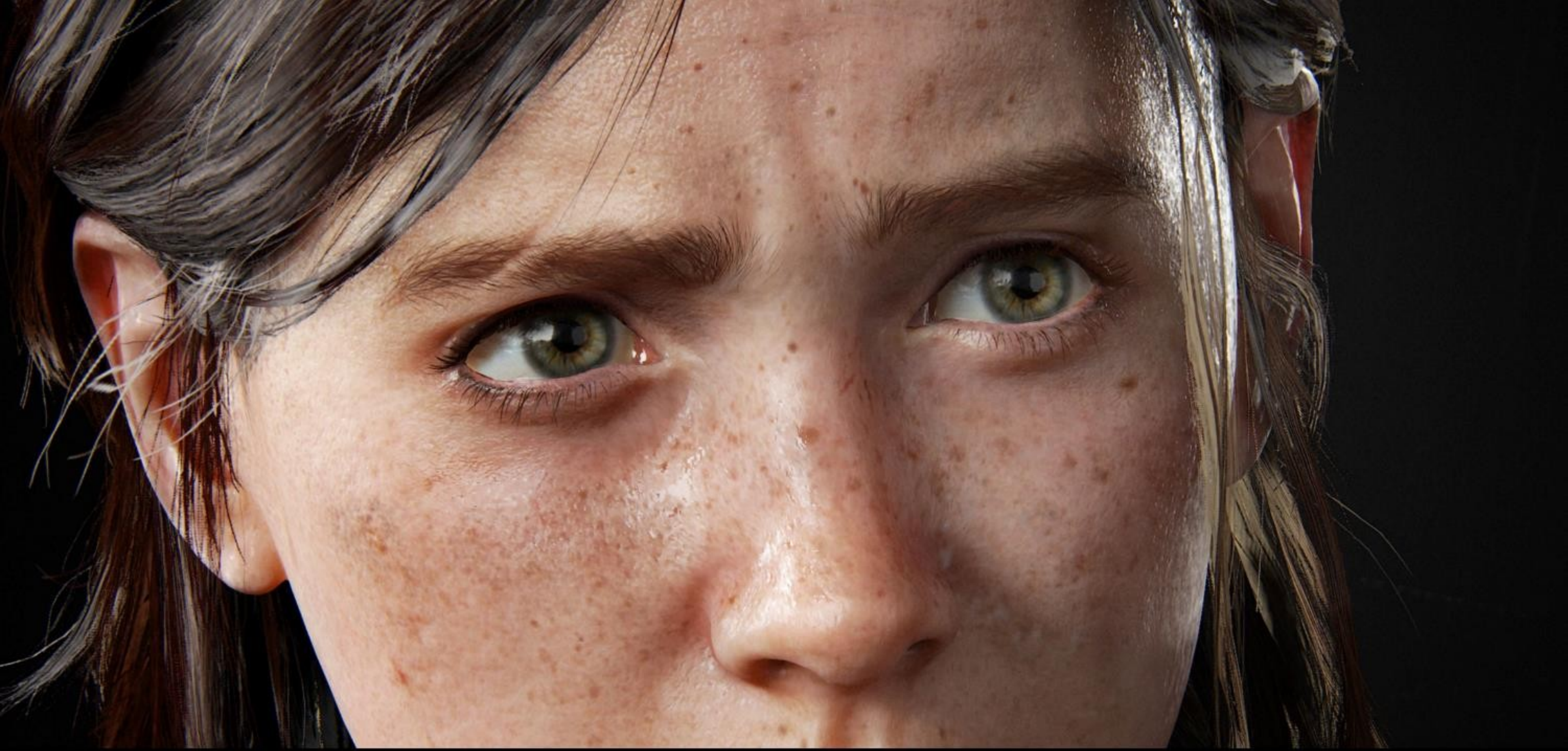


WETNESS ✓

WHAT IS “RIGHT”?

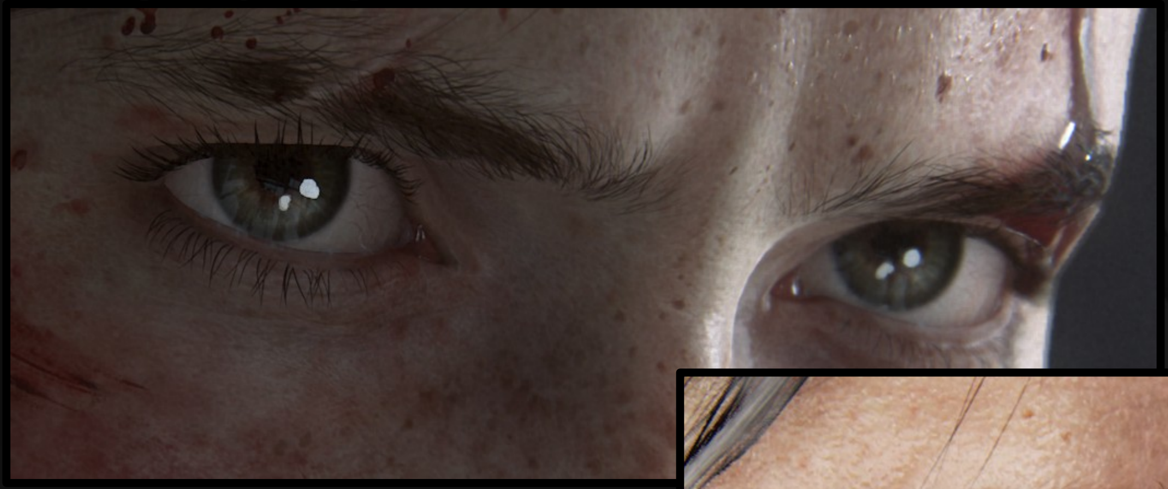

Profoto D1 @ 7.8
Dis:109" Height: 49"
120 degrees
Dis:50" Height: 92
140 degrees
Dis: 86" Height: 61"
Dis:115" Height:80"
50 degrees
Profoto D1 @ 7.3
Profoto D1 @ 5.3



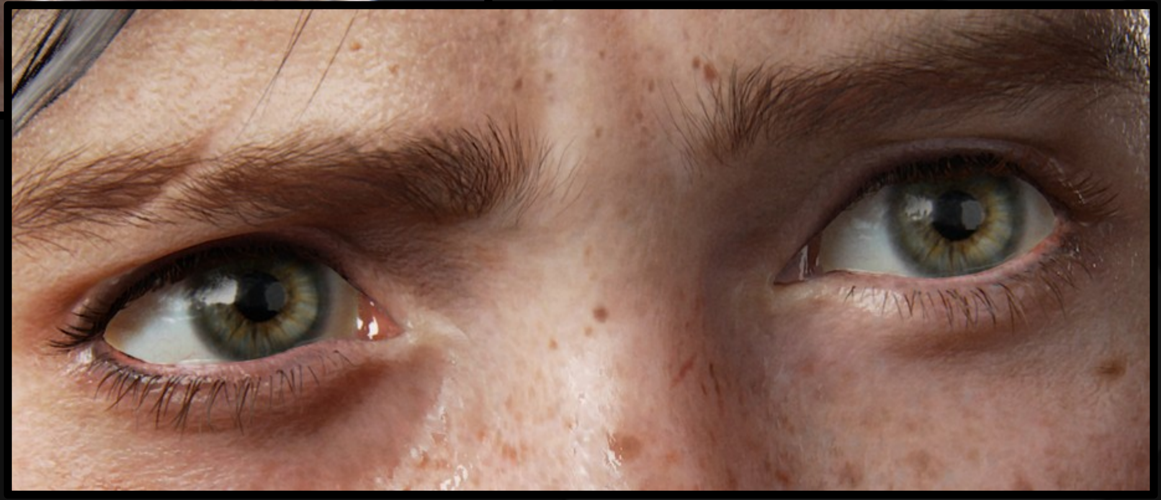


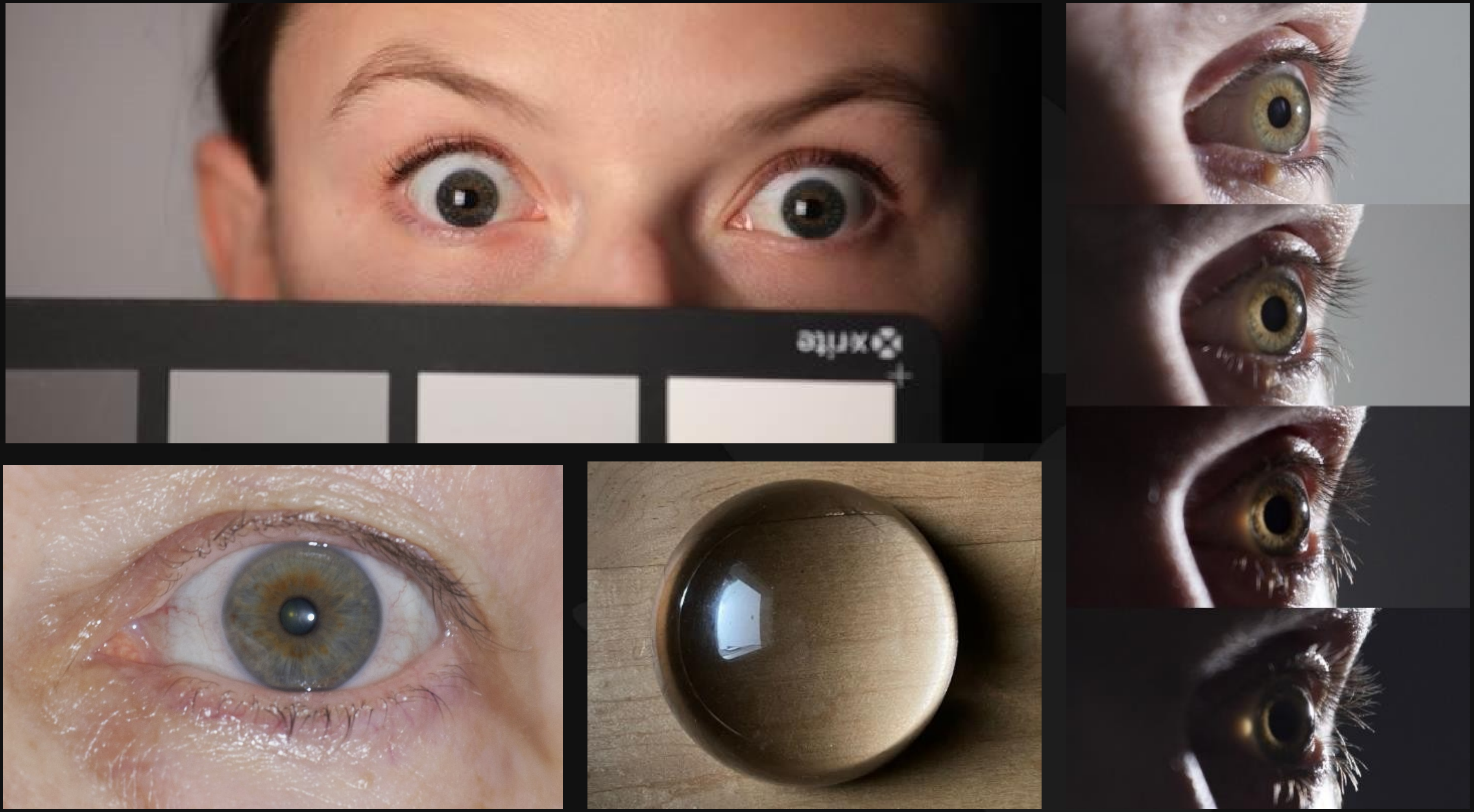
EYES

Before...



...After

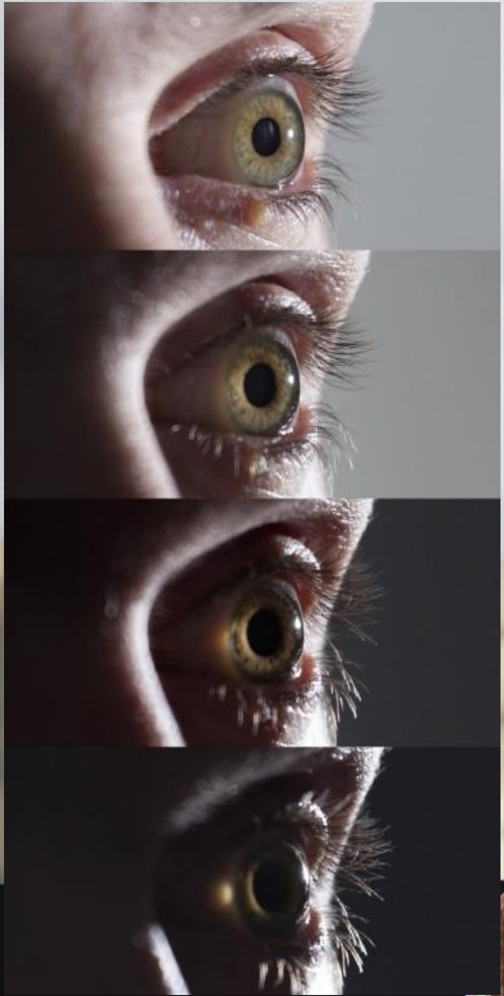


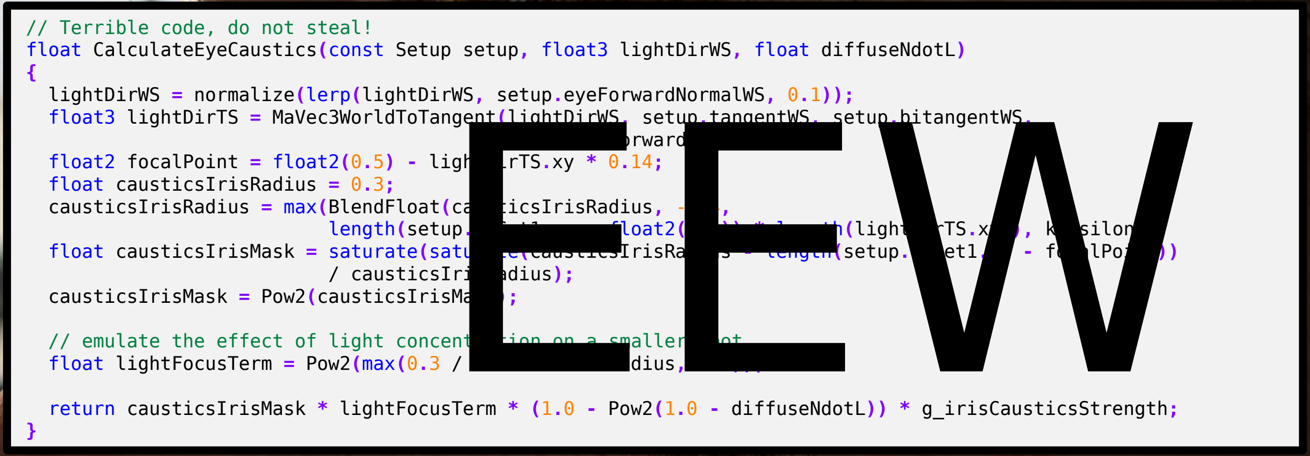

xrite

Shadows and Occlusion

IBL Occlusion

Caustics

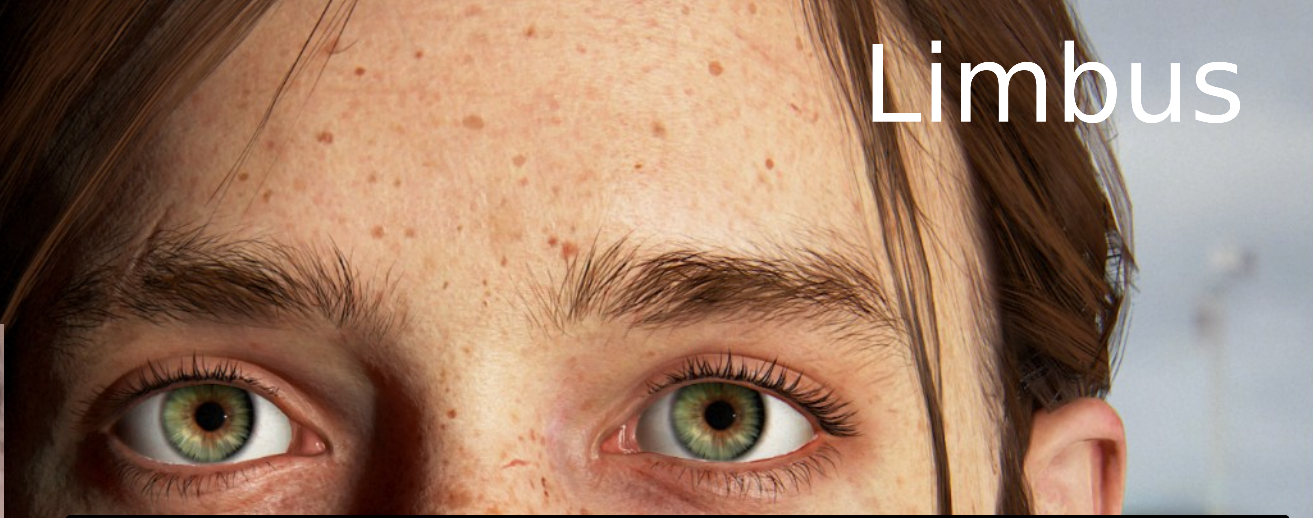


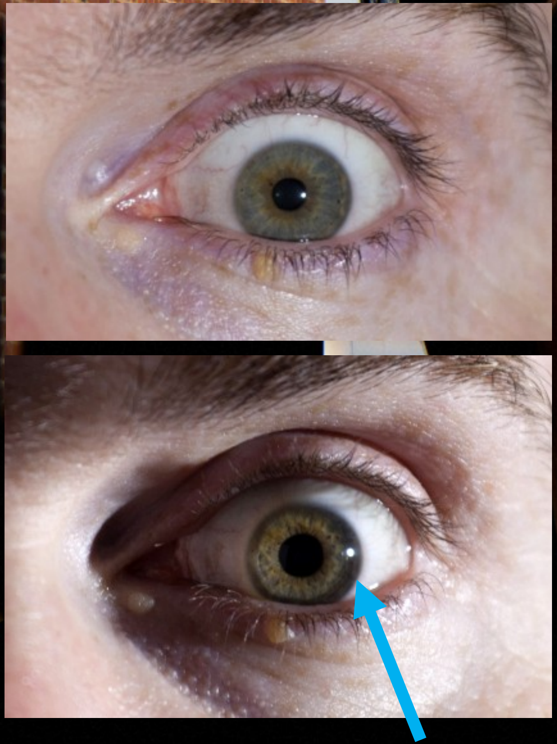


```
// Terrible code, do not steal!
float CalculateEyeCaustics(const Setup setup, float3 lightDirWS, float diffuseNdotL)
{
  lightDirWS = normalize(lerp(lightDirWS, setup.eyeForwardNormalWS, 0.1));
  float3 lightDirTS = MaVec3WorldToTangent(lightDirWS, setup.tangentWS, setup.bitangentWS,
                                          [illegible]rward[illegible]
  float2 focalPoint = float2(0.5) - ligh[illegible]rTS.xy * 0.14;
  float causticsIrisRadius = 0.3;
  causticsIrisRadius = max(BlendFloat(ca[illegible]icsIrisRadius, [illegible],
                           length(setup.[illegible]float2([illegible])) * [illegible](light[illegible]rTS.x[illegible]), k[illegible]silon[illegible]
  float causticsIrisMask = saturate(satu[illegible]e(causticsIrisRa[illegible]s - length(setup.[illegible]et1.[illegible] - fo[illegible]lPo[illegible]))
                           / causticsIri[illegible]adius);
  causticsIrisMask = Pow2(causticsIrisMa[illegible]);

  // emulate the effect of light concent[illegible]ion on a smaller[illegible]ot
  float lightFocusTerm = Pow2(max(0.3 / [illegible]dius,[illegible]));

  return causticsIrisMask * lightFocusTerm * (1.0 - Pow2(1.0 - diffuseNdotL)) * g_irisCausticsStrength;
}
```

EEW

Limbus

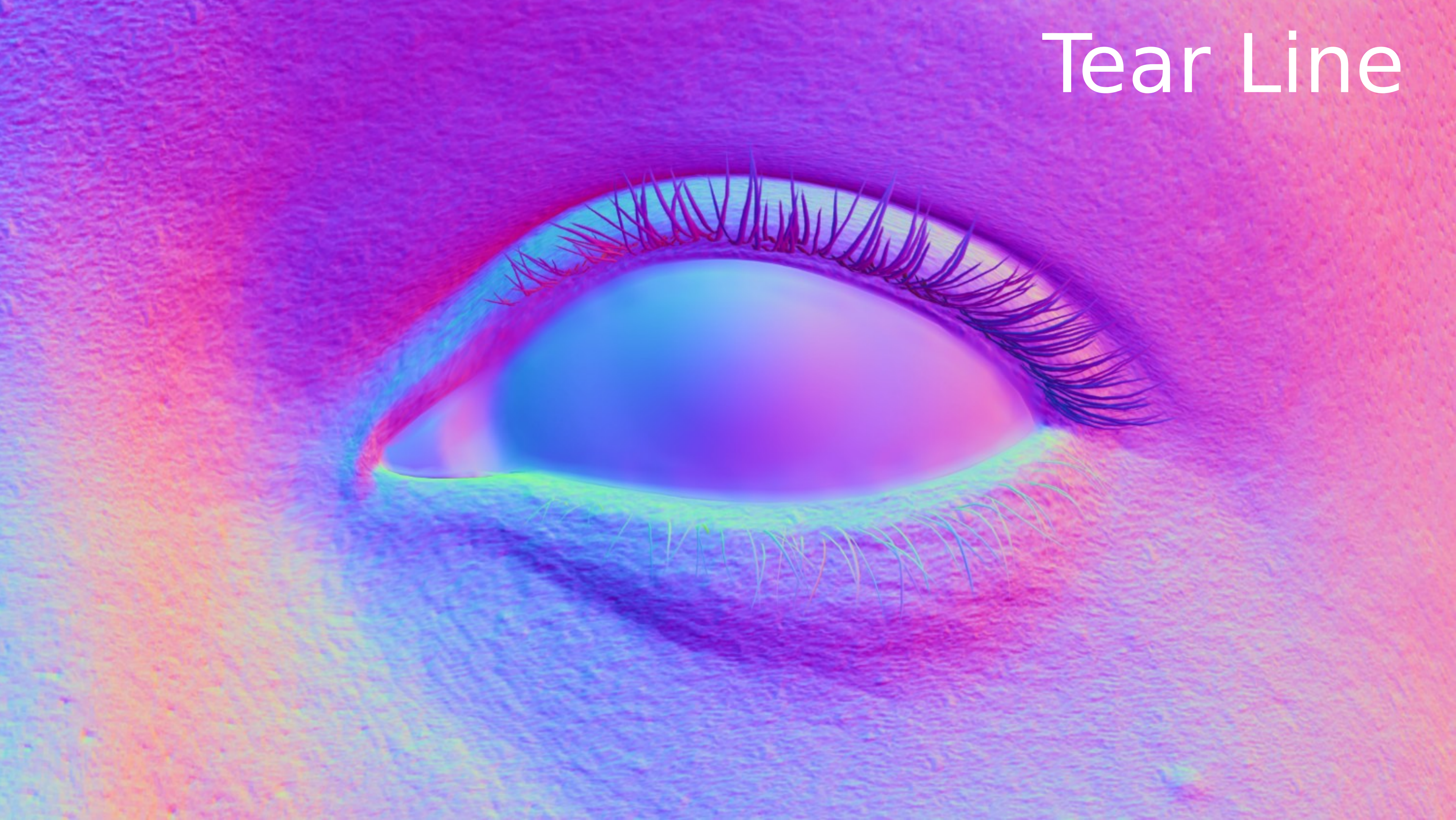




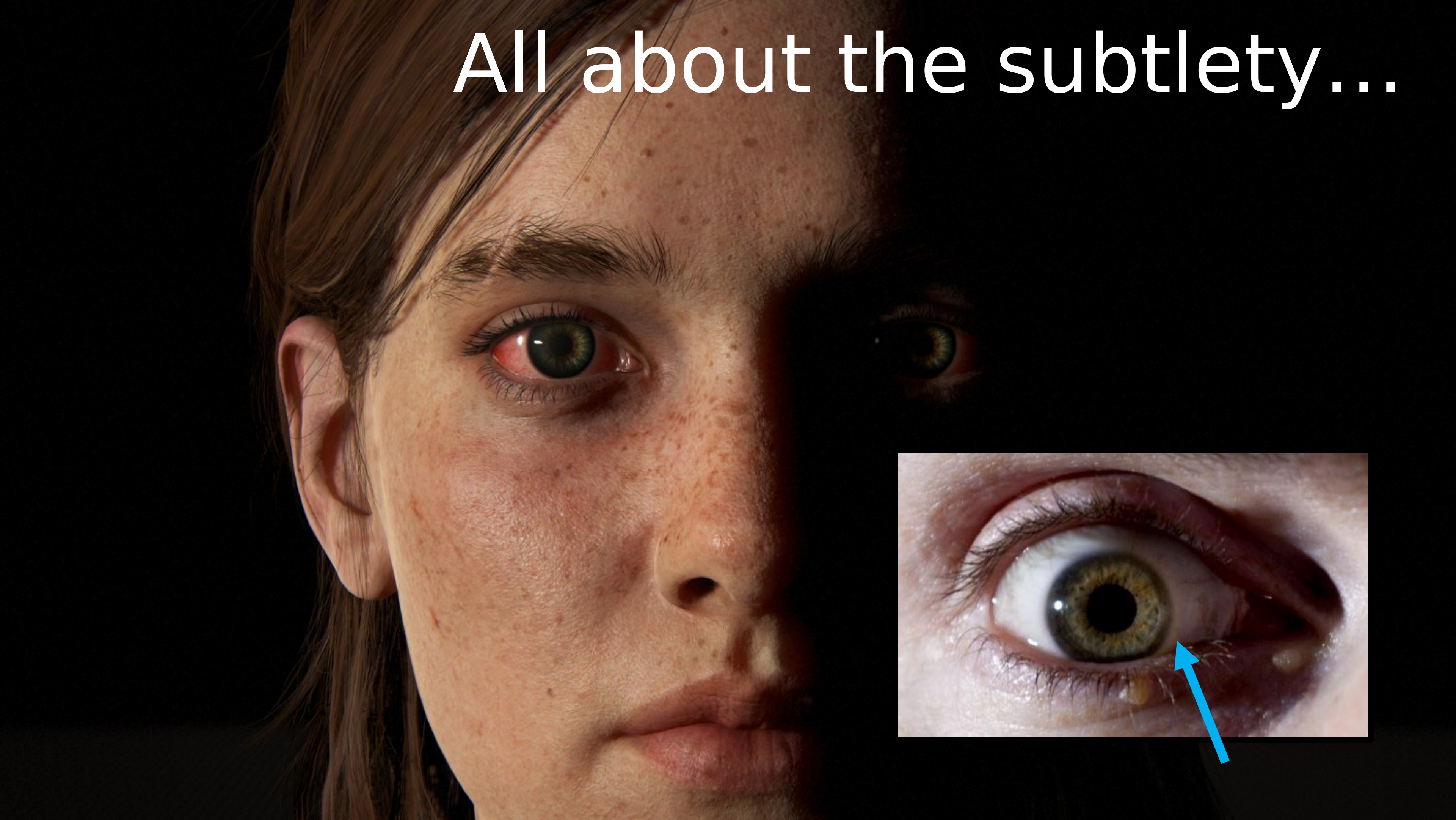
```
// Another hack!  Do not steal!
float CalculateEyeLimbus(float distanceToIris, float NdotL)
{
  // Constants are a standardized distance in UV space from the
center
  // of the pupil
  float refractionScalar = 0.03  (1  - NdotL);
  float limbusEdge = kScleraCenter -  .015   refractionScalar
  float limbusInner = limbusEdge - 0. 45 - refractionScalar;

  // Pure artistic tuning
  float limbus = smoothstep(limbusEdge, limbusInner,
distanceToIris);
  limbus = lerp(limbus, Pow4(limbus), g_limbusStrength * 2.0 -
1.0);
  return limbus;
```

UGH

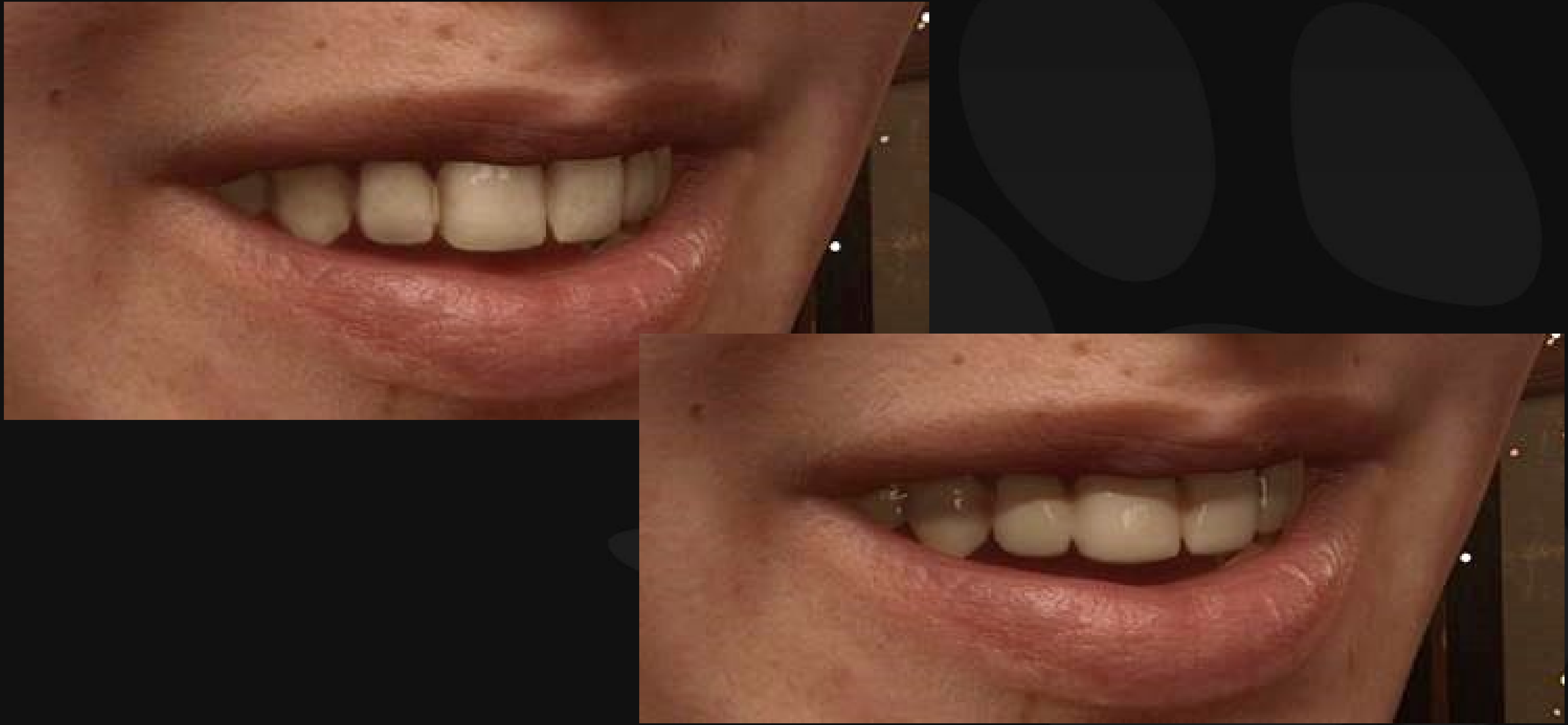
Tear Line

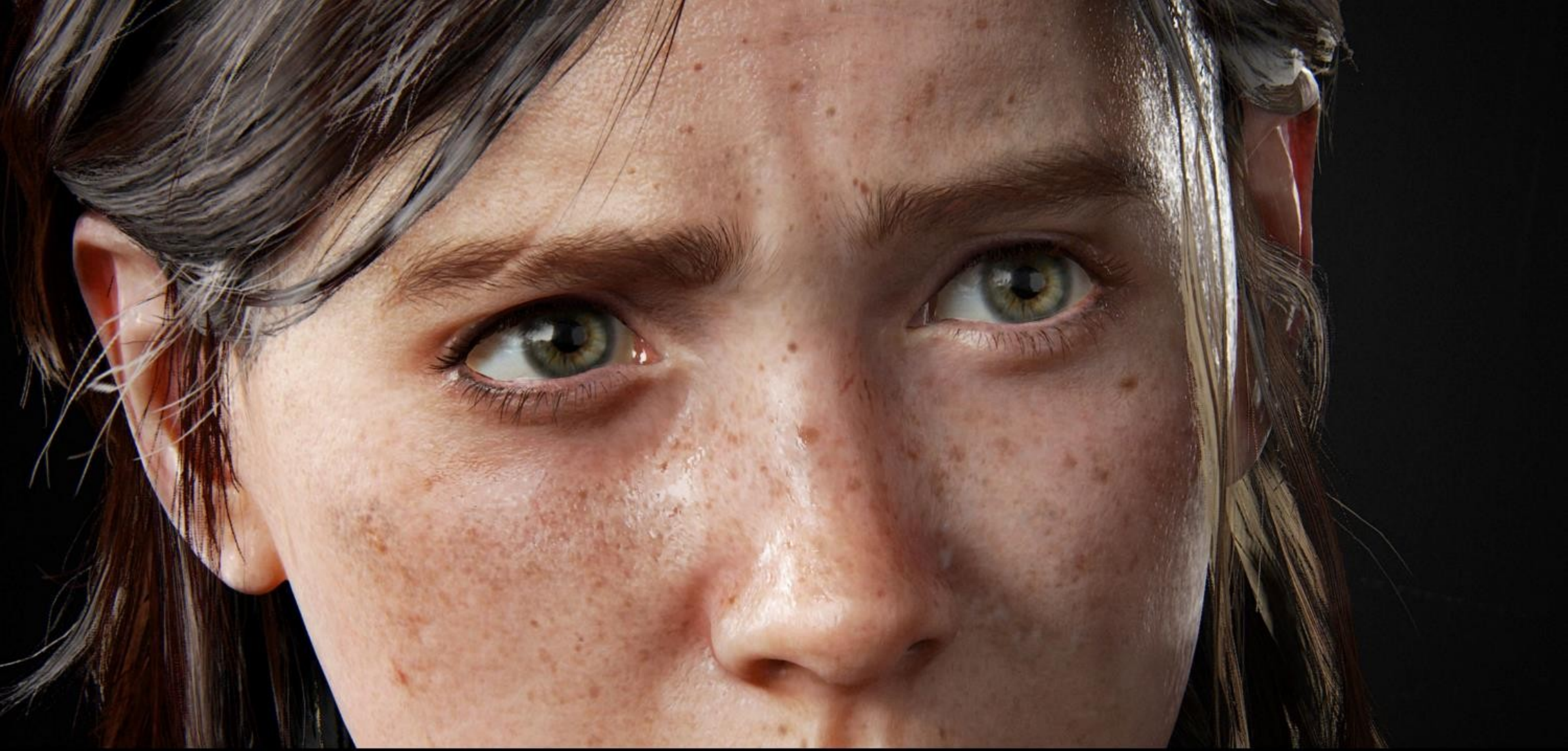


All about the subtlety...



Teeth



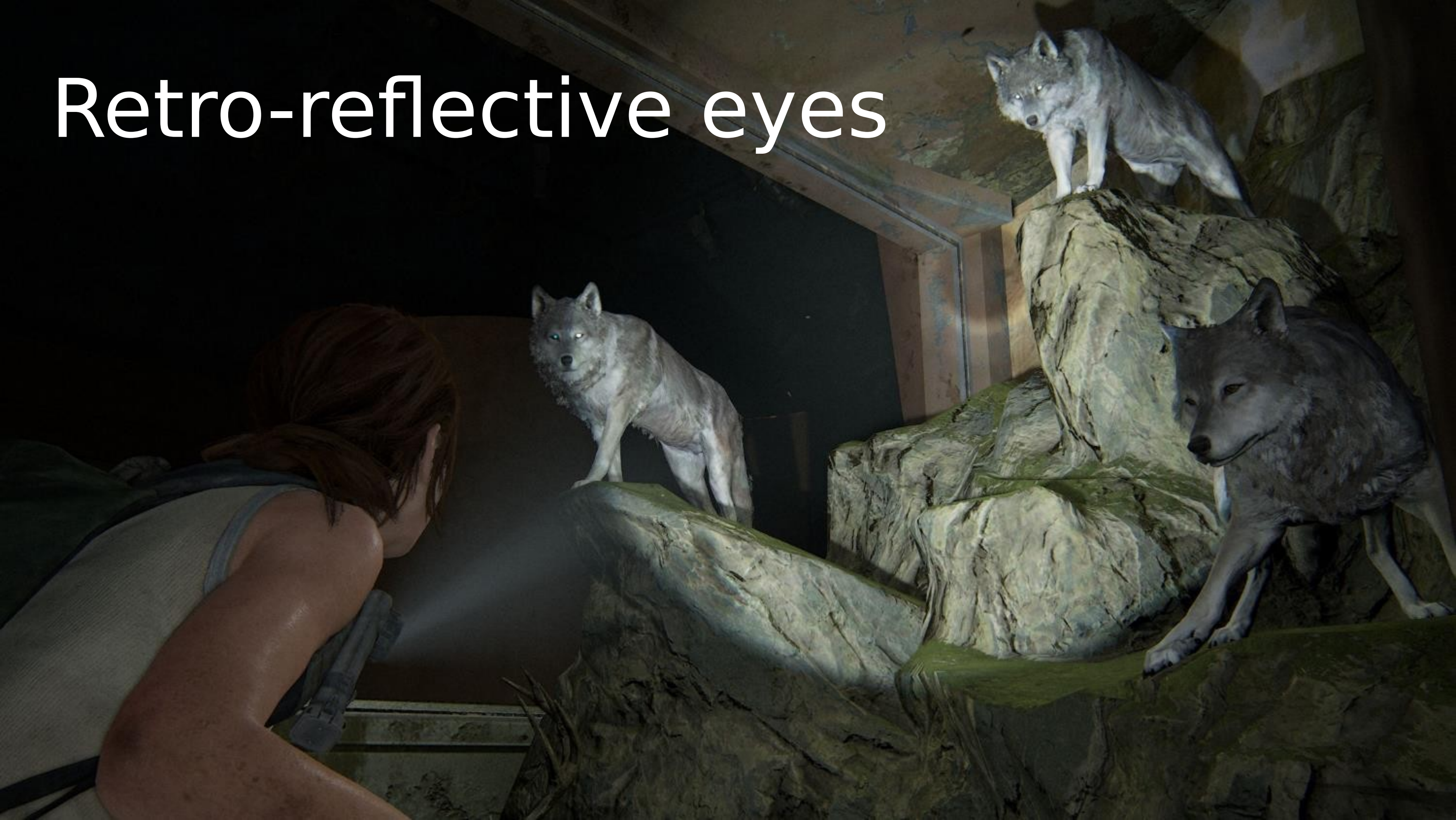


EYES ✓

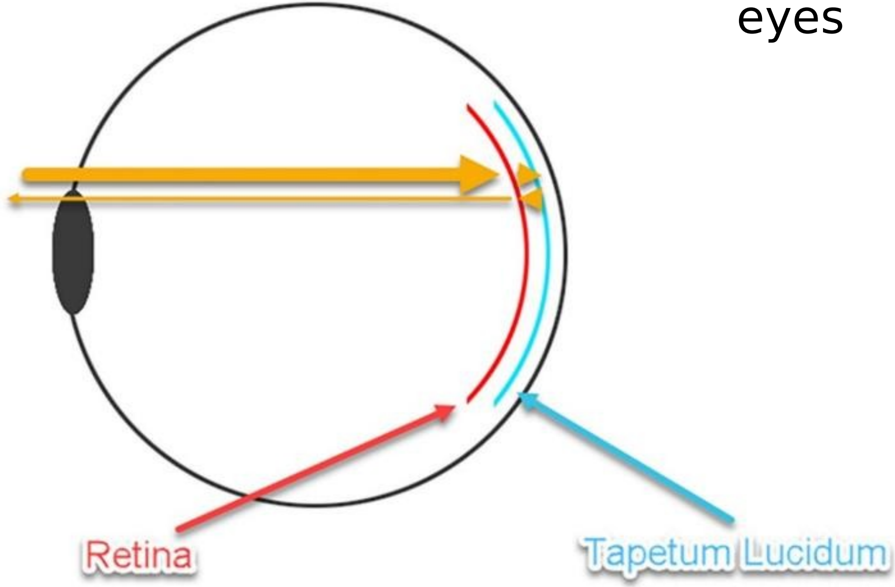
STEVEN
TANG

Technical Artist / Naughty Dog

Retro-reflective eyes

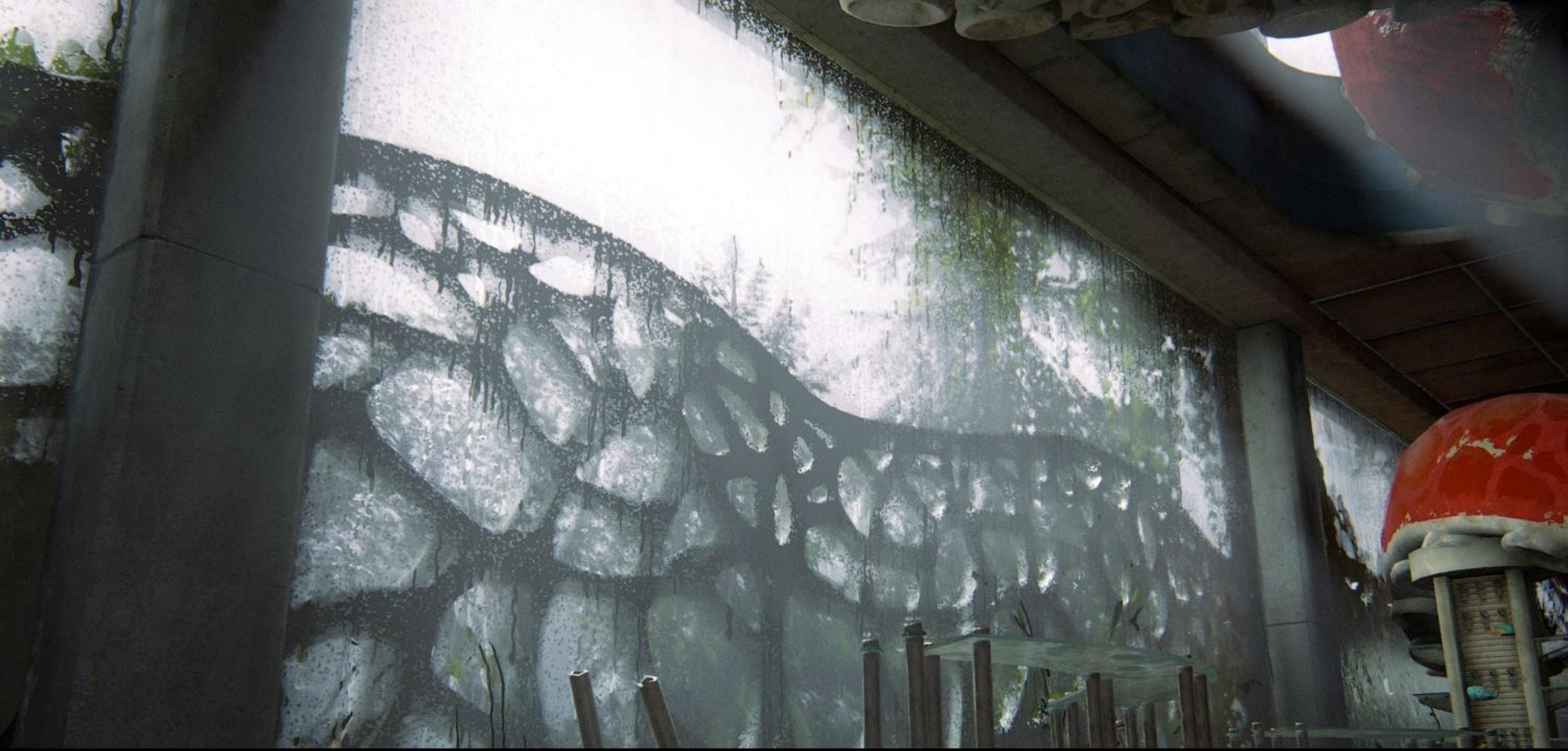


Retro-reflective eyes


Retina
Tapetum Lucidum

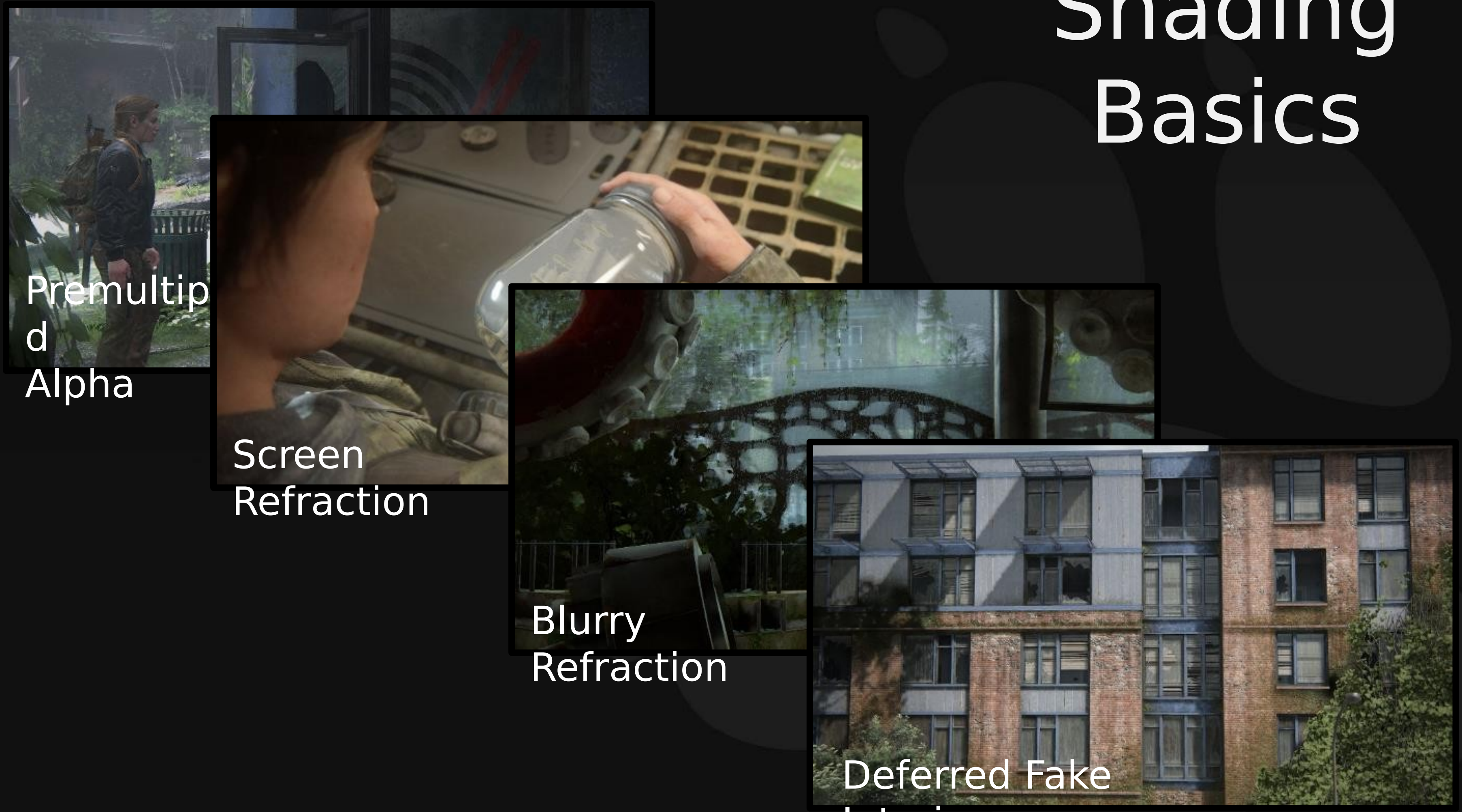
```
// retroreflectiveEyeTint is an artist supplied parameter
float3 CalculateEyePupilReflection(float3 lightDirWS, float3 viewDirWS, float3 retroreflectiveEyeTint, float3 eyeNormalWS)
{
    float3 L = lightDirWS;
    float3 V = viewDirWS;
    float3 H = normalize(L + V);
    float3 N = lightDirWS;

    // reuse the standard specular BRDF function
    float3 finalSpec = SpecularBRDF(L, V, N, H, baseColor=retroreflectiveEyeTint, metallic=1.0, specular=0.5,
roughness=0.1);

    // more on thin film interference later
    finalSpec = ApplyThinFilmInterference(finalSpec, L, eyeNormalWS, [some hard-coded parameters tuned against reference]);

    finalSpec *= FetchIrisPupilMask();

    return finalSpec;
}
```

EYES ✓



GLASS

Shading Basics



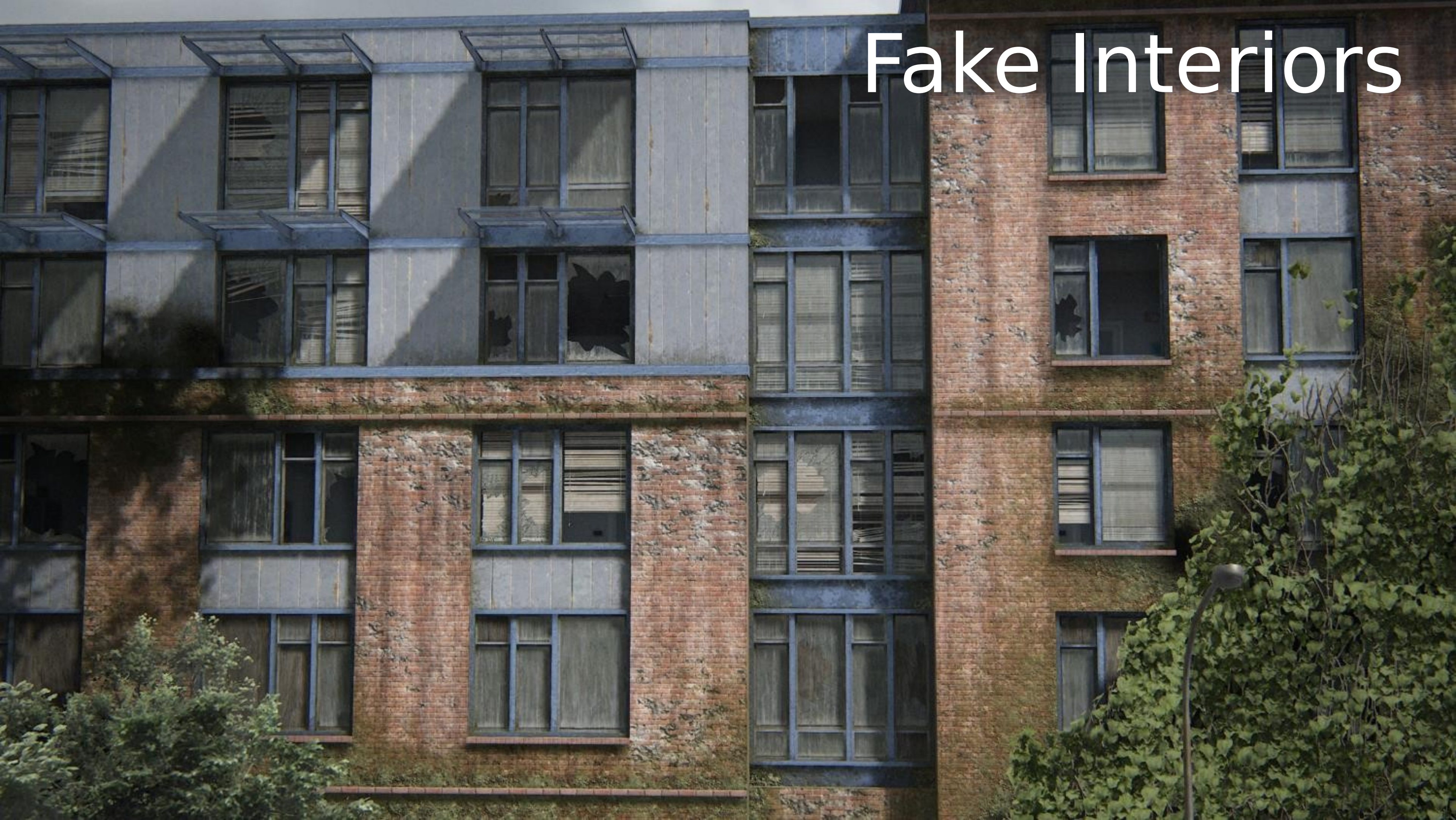
Premultipd Alpha

Screen Refraction

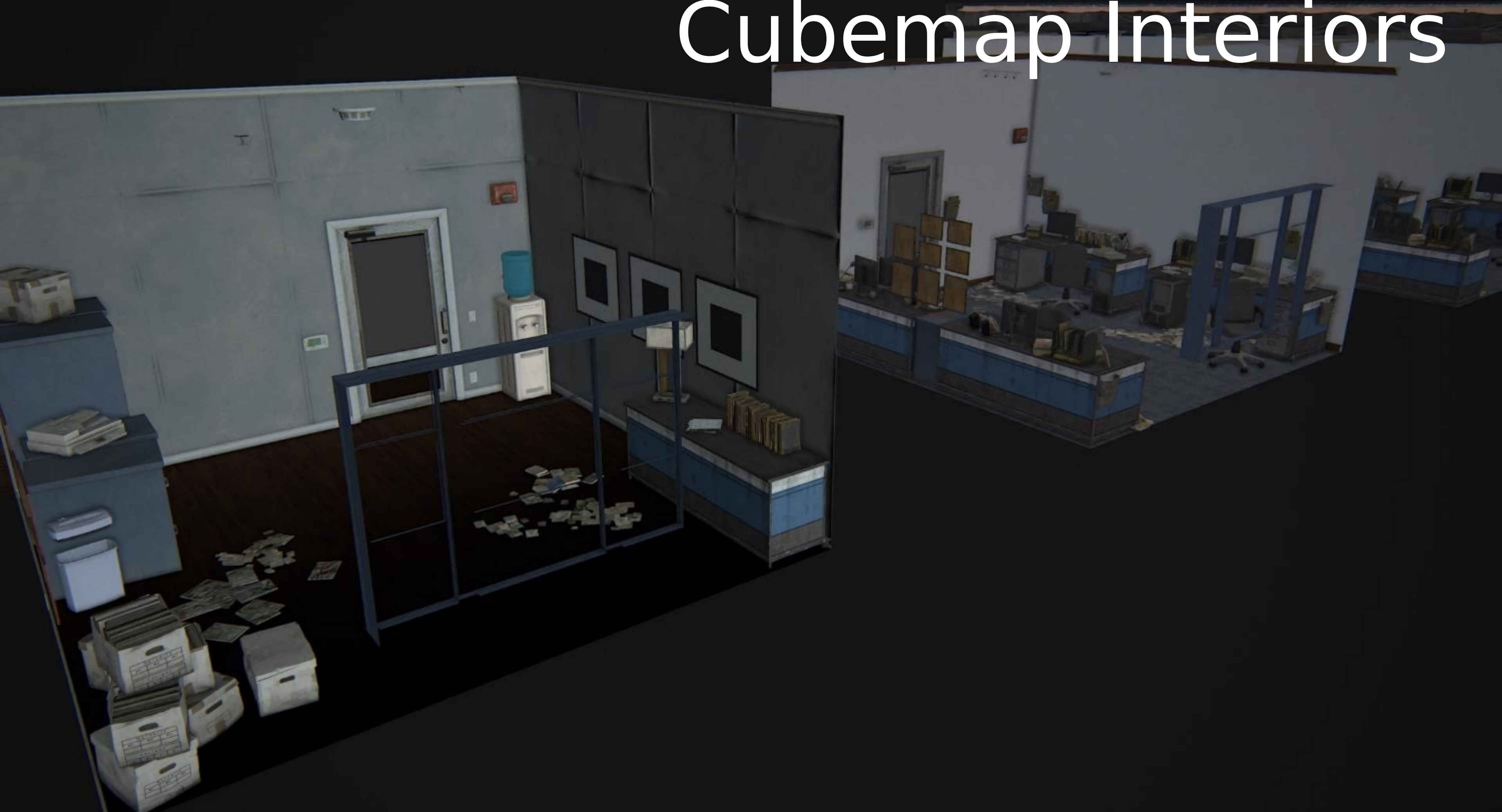
Blurry Refraction

Deferred Fake Interiors

Fake Interiors

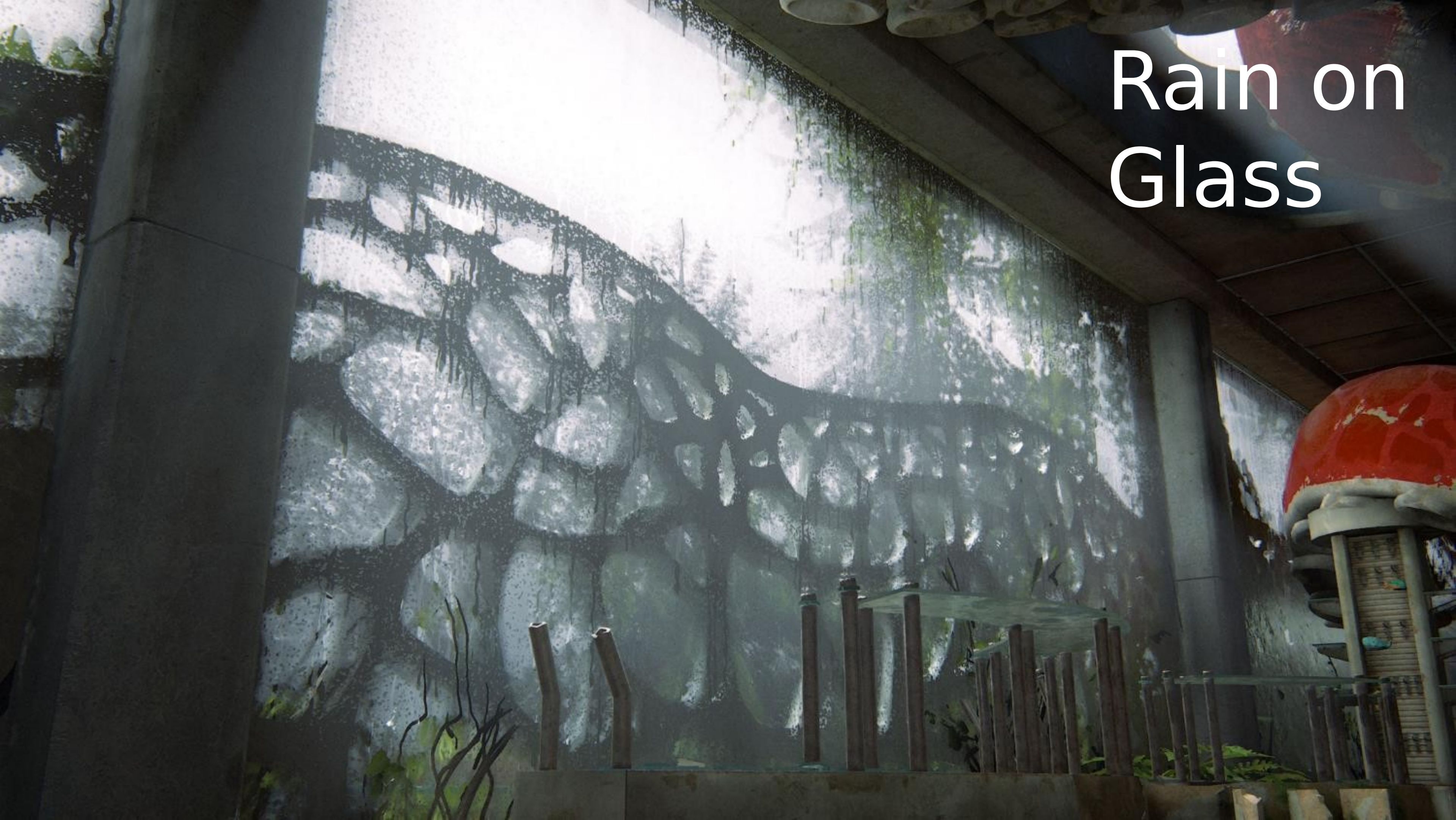


Cubemap Interiors



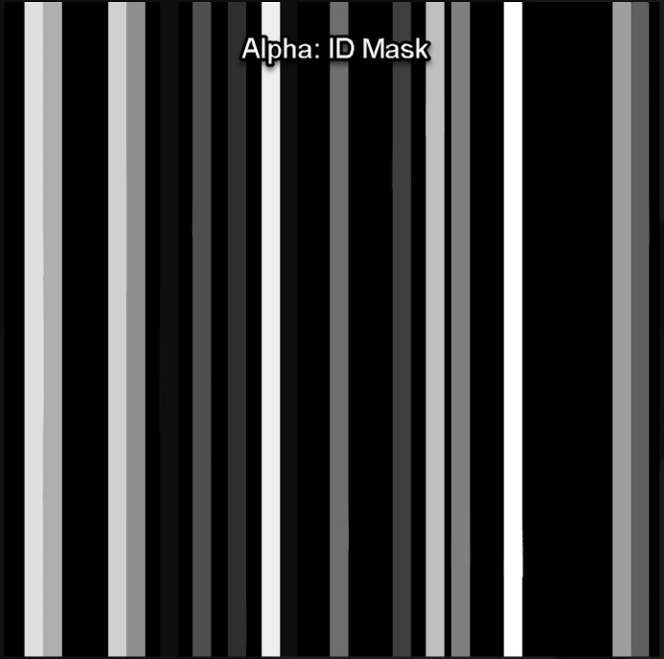
```
float4 GetParallaxCubemapInteriorCoords(const Setup setup, float cubeDepth, float3 perspectiveCorrection)
{
// Setup
float3 cameraPosWS = setup.cameraPositionWS;
float3 positionWS = setup.positionWS;
float3 N = setup.vertexNormalWS;
float3 T = setup.tangentWS;
float3 B = setup.bitangentWS;
float3 V = setup.viewWS;
float3 vT = T * sign(dot(V, T));
float3 vB = B * sign(dot(V, B));
float2 uv = setup.uvSet1.uv - float2(0.5);
float3 cubeDimensions = float3(0.5, 0.5, 0.5 * cubeDepth);

// Find points on the cube
float3 frontPlaneCenterWS = positionWS - T * uv.x - B * uv.y;
float3 cubeCenterWS = frontPlaneCenterWS - N * cubeDimensions.z;
float3 backPlaneCenterWS = frontPlaneCenterWS - N * cubeDimensions.z * 2.0;
float3 sidePlaneCenterWS = cubeCenterWS - vT * cubeDimensions.x;
float3 topPlaneCenterWS = cubeCenterWS - vB * cubeDimensions.y;

// Find intersections with the planes
float3 backIntersectionWS = LinePlaneIntersectionWS(N, backPlaneCenterWS, V, positionWS);
float3 sideIntersectionWS = LinePlaneIntersectionWS(vT, sidePlaneCenterWS, V, positionWS);
float3 topIntersectionWS = LinePlaneIntersectionWS(vB, topPlaneCenterWS, V, positionWS);

// Choose which plane is closest
float lenBack = LengthSquared(backIntersectionWS - cameraPosWS);
float lenSide = LengthSquared(sideIntersectionWS - cameraPosWS);
float lenTop = LengthSquared(topIntersectionWS - cameraPosWS);

float3 intersectionWS = lenBack < lenSide ? backIntersectionWS : sideIntersectionWS;
intersectionWS = (lenTop < lenBack && lenTop < lenSide) ? topIntersectionWS : intersectionWS;

// Transform to cube space
float3 intersectionCS = intersectionWS - cubeCenterWS;
intersectionCS = float3(dot(intersectionCS, T), dot(intersectionCS, B), dot(intersectionCS, N));
intersectionCS /= cubeDimensions;

intersectionCS *= perspectiveCorrection;

float pixelDepth = min3(lenBack, lenSide, lenTop);

return float4(intersectionCS, pixelDepth);
}
```

Rain on Glass



window-rain-a.tga

Animating Droplets


Alpha: ID Mask

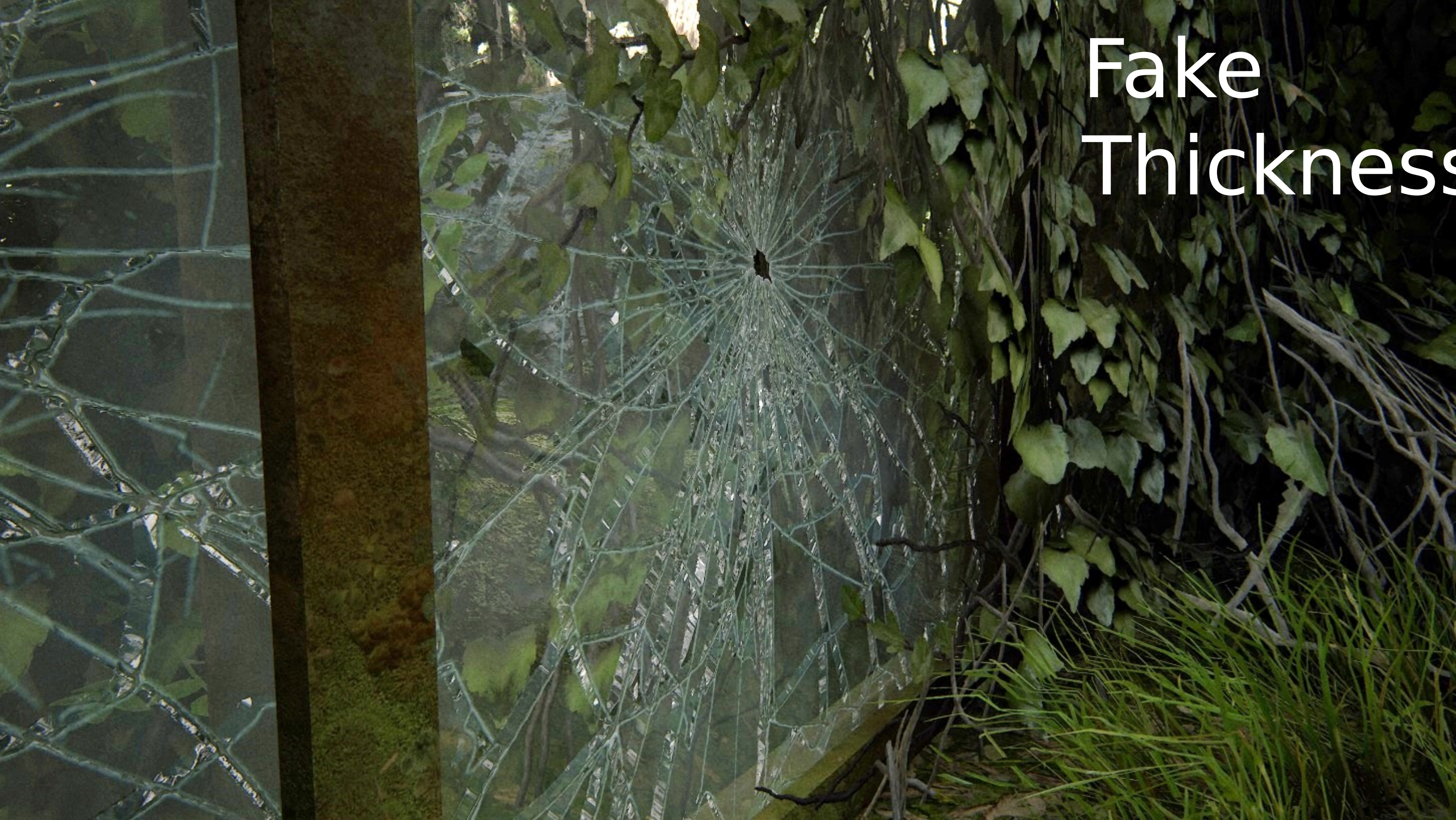
Animating Droplets

Time Variance

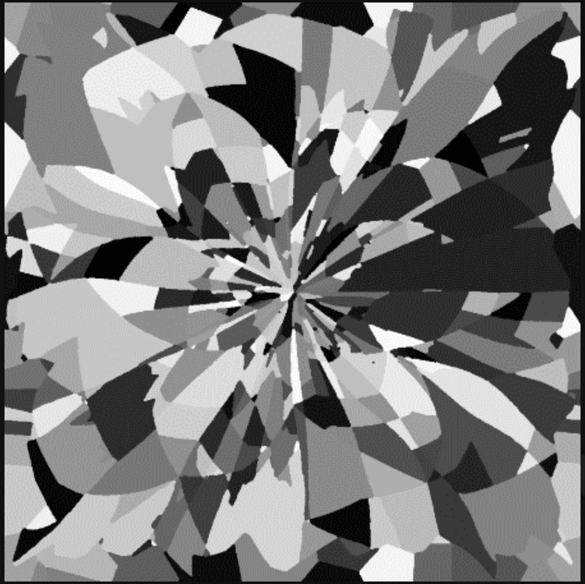
- Droplets do not travel at uniform speed

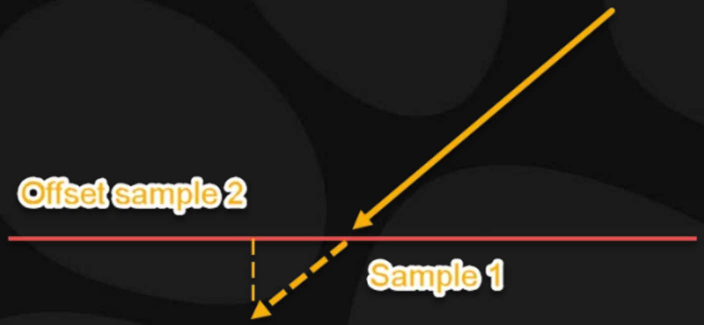
```
float GetMonotonicRandomizedWindowRainTime(float offset, float speedRandomness, float time)
{
    // artist provide speedRandomness parameter
    // offset parameter is determined by the ID mask mentioned above
    // the coefficients are magic numbers tuned to look right. We came up with a function first, then remapped it
    // to above 0 range and took integral to guarantee monotonically increasing behavior, so the rain droplets
    // never travel backwards.
    float noisedTime = (6.4 * time + 1.4 * sin(2.5 * (time + offset)) + 0.205 * sin(7.3 * (time + offset))
                        + 0.081 * sin(12.3 * (time + offset))) * 0.167;

    return lerp(time, noisedTime, saturate(speedRandomness));
}

…
uvPosition.v = GetMonotonicRandomizedWindowRainTime(offset, speedRandomness, time) * dropletSpeed + offset;
…
```

Fake Thickness



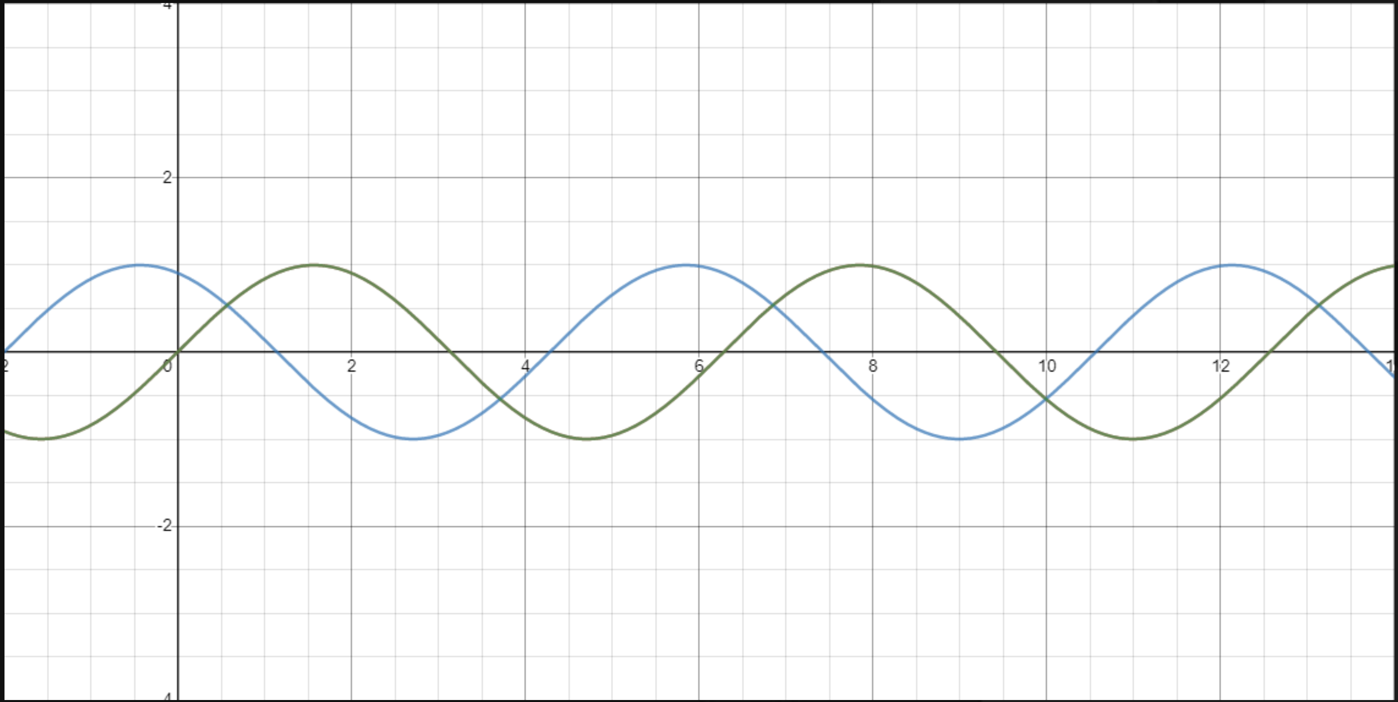
- Relatively cheap and easy fake thickness
  - 4 texture samples + simple math



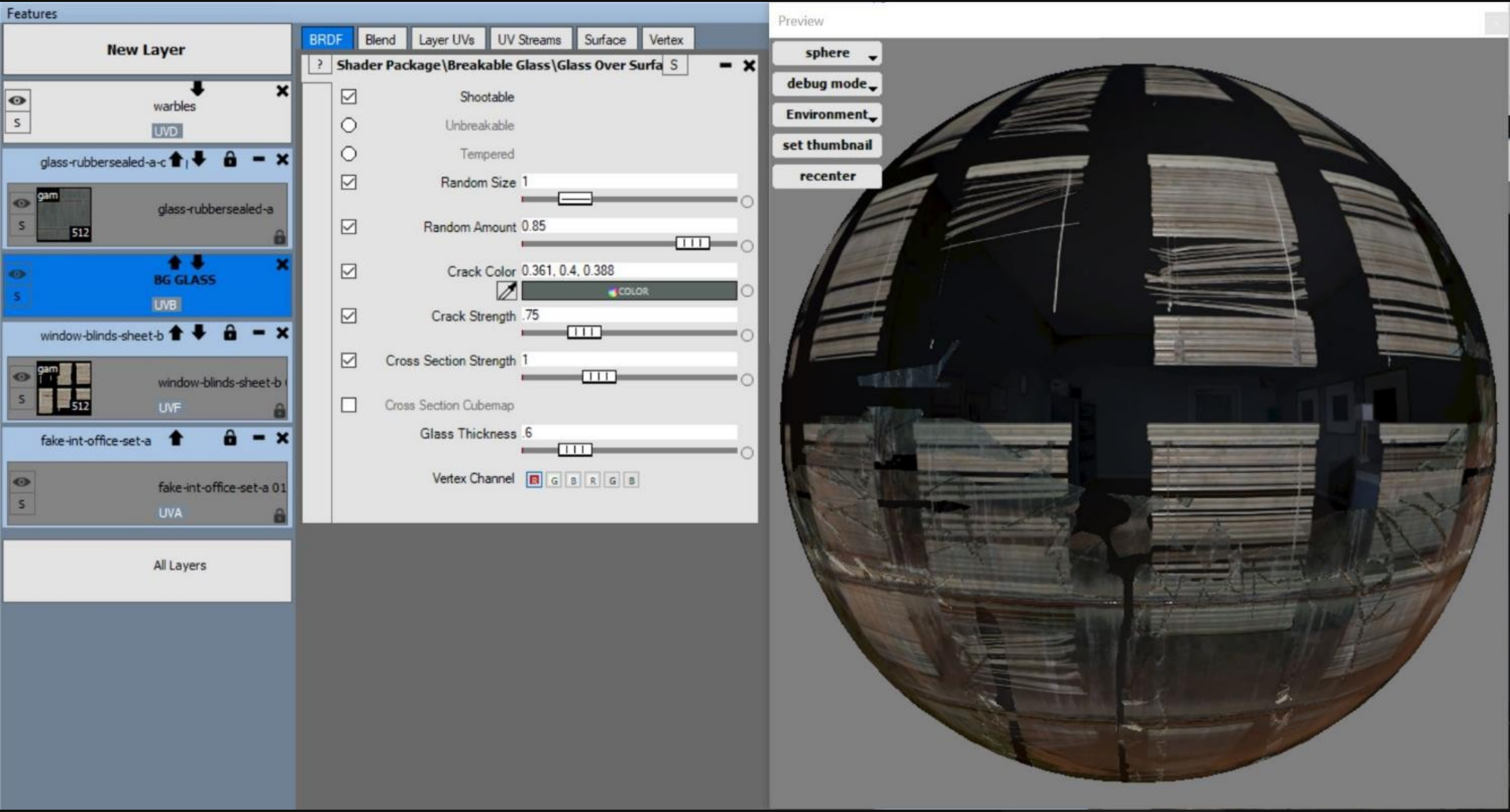

Offset sample 2
Sample 1

Thin Film Interference

Thin film interference



```
// specularLight is the regular specular coming from the specular BRDF. filmThickness uses nanometer as unit. filmStrength is an artist controlled intensity.
float3 SeemsLegitThinFilm(float3 specularLight, float filmThickness, float filmStrength, float NdotL, bool doublePhase=false)
{
    // The values we use for wavelength of red, green and blue lights
    const float3 rgbLightWavelength = float3(700.0, 510.0, 440.0);

    // Ignoring refraction for simplified math
    float extraDistance = filmThickness / NdotL;

    float3 phaseChangeEquivalentShift = 0.0;
    // If the film has higher ior than glass, the reflection phase change happens twice, which shifts the phase by a full wavelength and we don’t need to worry about it
    // Otherwise, it’s shifted by half the wave length.
    if (!doublePhase)
        phaseChangeEquivalentShift = - rgbLightWavelength * 0.5;

    float3 phaseOffset = float3(extraDistance - phaseChangeEquivalentShift) / rgbLightWavelength;
    float3 superpositionAmplitude = abs(cos(phaseOffset * Pi));

    // Using 5 femtoseconds as the coherence time of sunlight, the calculated coherence distance using speed of light is about 1500nm
    float coherenceTerm = 1.0 - saturate(extraDistance * 1.0 / 1500);

    superpositionAmplitude = lerp(1.0, superpositionAmplitude, coherenceTerm);

    // The physical justification for filmStrength is how much light is absorbed by the film itself. It is  multiplied by NdotL since it should scale with the film thickness.
    float filmStrengthTerm = filmStrength * NdotL;
    specularLight = lerp(specularLight, specularLight * superpositionAmplitude, filmStrengthTerm);

    return specularLight;
}
```


Features
New Layer
warbles
UVD
glass-rubbersealed-a-c
gam
512
glass-rubbersealed-a
BG GLASS
UVB
window-blinds-sheet-b
gam
512
window-blinds-sheet-b
UVF
fake-int-office-set-a
fake-int-office-set-a 01
UVA
All Layers
BRDF
Blend
Layer UVs
UV Streams
Surface
Vertex
Preview
sphere
debug mode
Environment
set thumbnail
recenter
COLOR

Shader Package\Breakable Glass\Glass Over Surfa

☑ Shootable

○ Unbreakable

○ Tempered

☑ Random Size 1

☑ Random Amount 0.85

☑ Crack Color 0.361, 0.4, 0.388

☑ Crack Strength .75

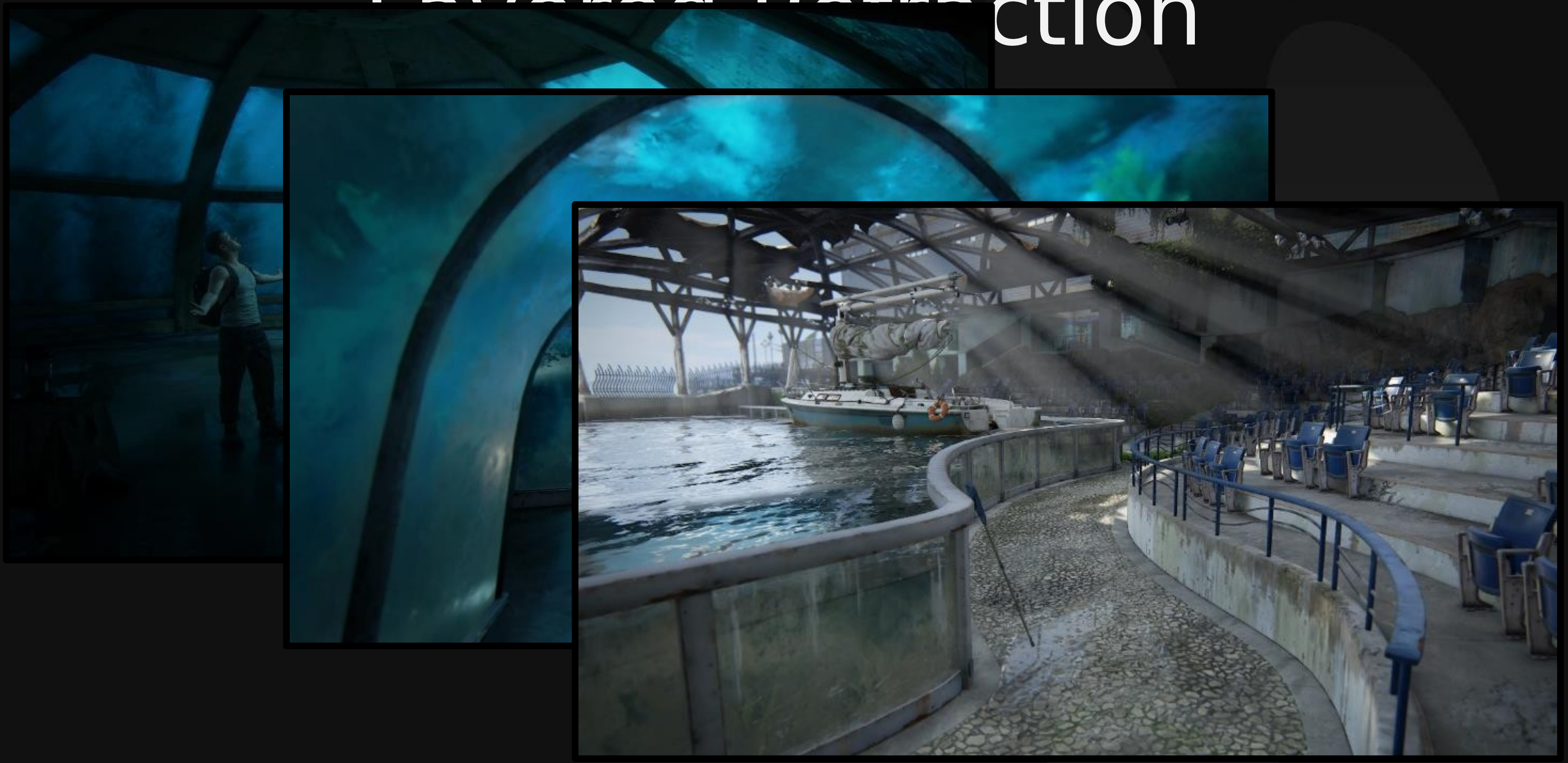
☑ Cross Section Strength 1

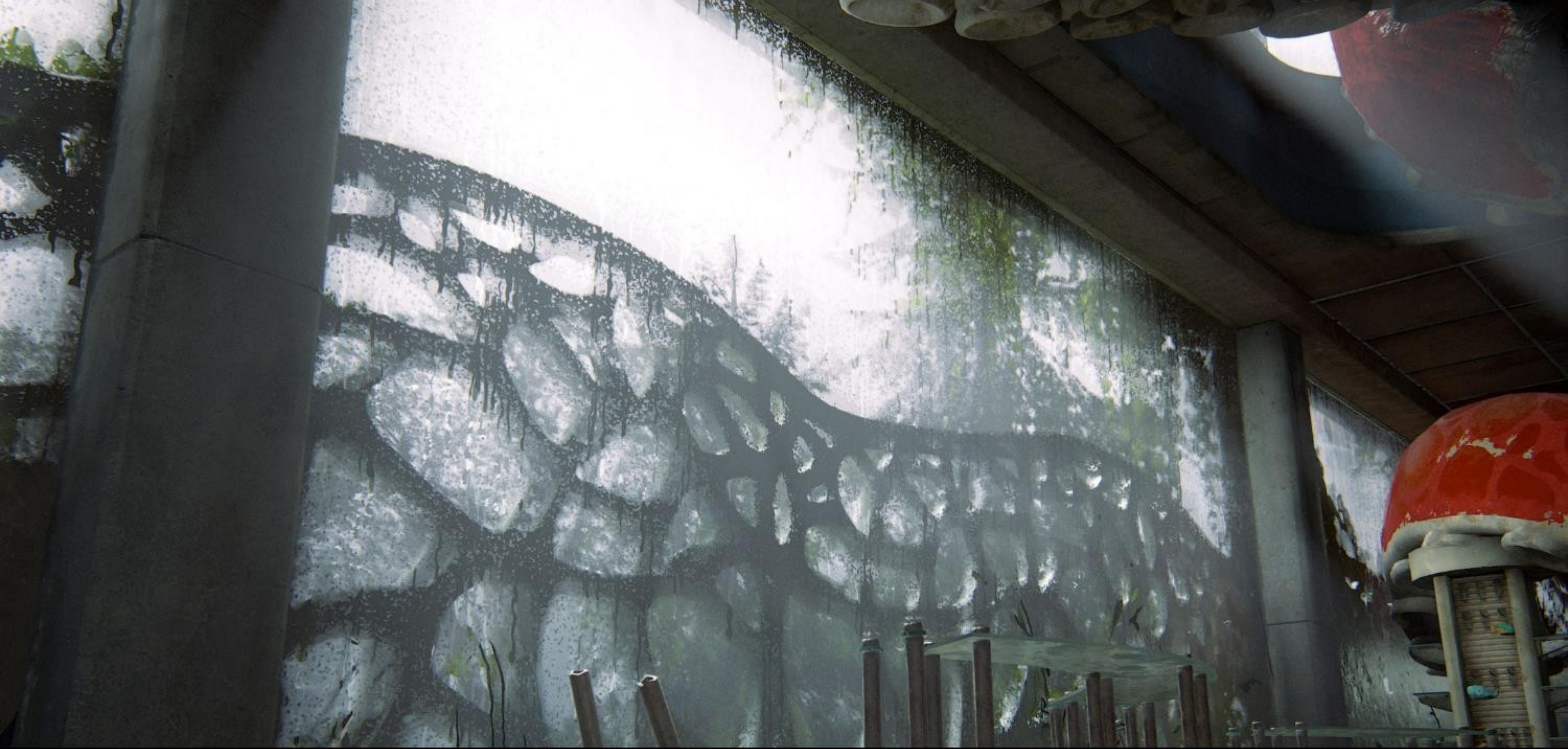
☐ Cross Section Cubemap

Glass Thickness .6

Vertex Channel R G B R G B

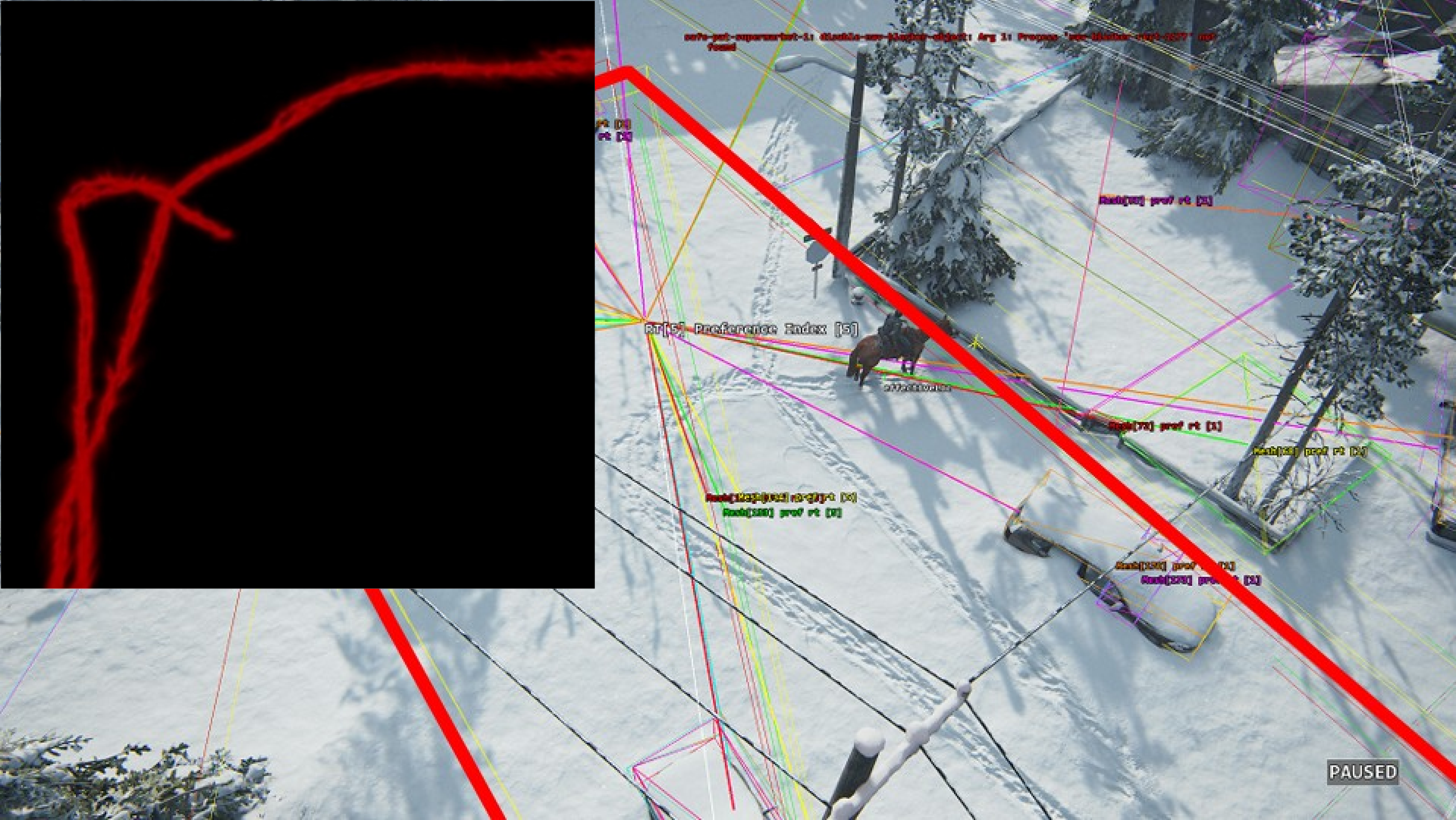
Blurry Refraction / Multi-Layered Refraction





GLASS ✓

SNOW


Preference Index [5]
PAUSED

Getting Creative

“It’s all systemic, so we get things like this for free.”

- Eben Cook, Lead Fx Artist

SNOW ✓

What’s Next?

Summary of lessons learned

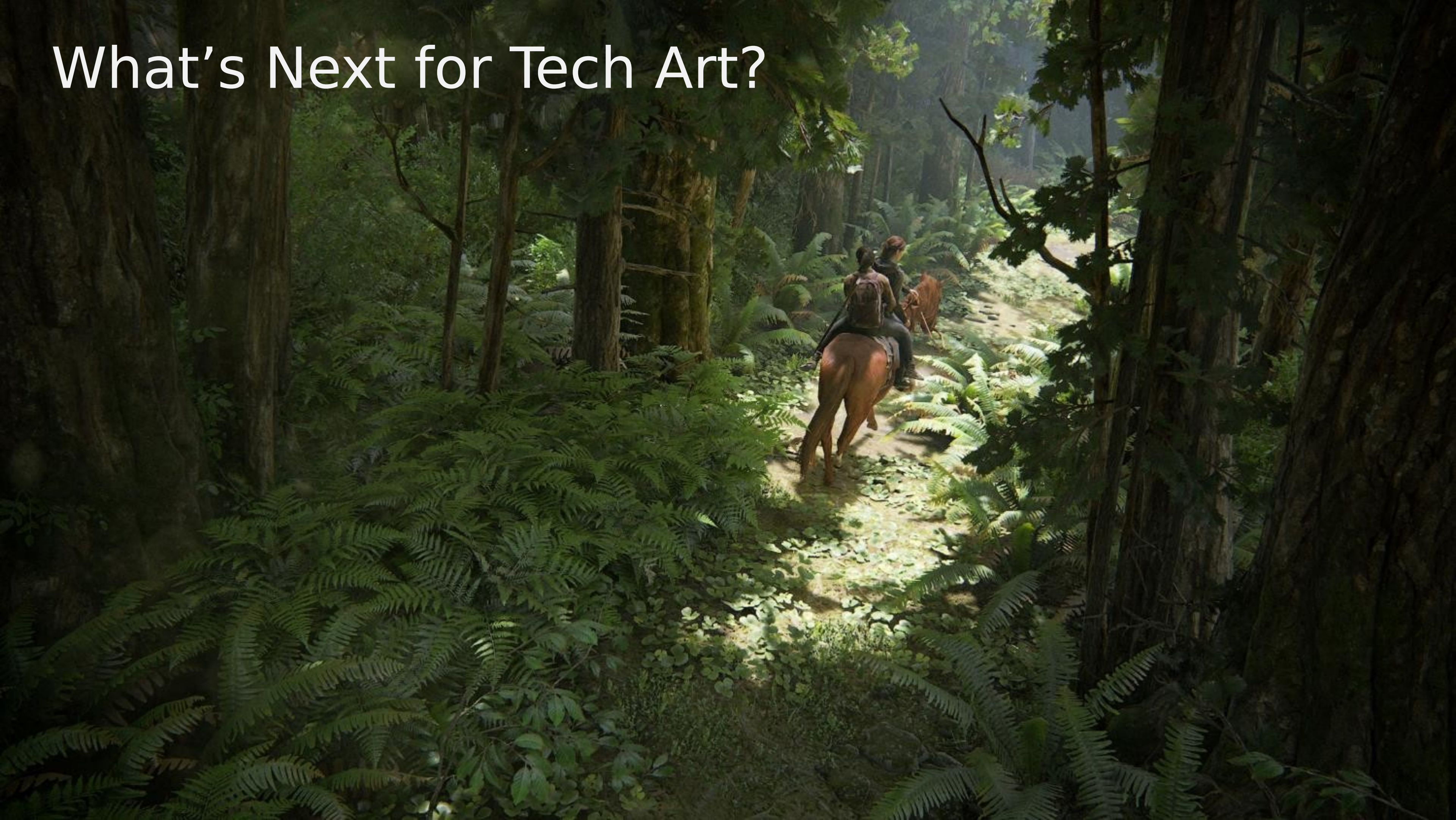
Recreate physical phenomena

Write good code

Elevate the team!

Be a bridge!

What’s Next for Tech Art?



Acknowledgements

The Naughty Dog Graphics Programmers:

Vincent Marxen, Abhishek Arora, Artem Kovalovs, Hawar Doghramachi, Jiayin Cao, Joe Forte, Ke Xu, Marshall Robin, Parikshit Saraswat, and Stephen Merendino, for all of the guidance they've given us, their patience… and for the incredible engine they've given us to build upon!

Acknowledgements

The Tech Art Team:

Lauren Garcia, Dongsub Woo, Kaleb Goulding, Wen Tien, and Barry Evans, whose additional contributions to our rendering code were immeasurable.

The VFX Team:

In particular Eben Cook, Raymond Popka, Wataru Ikeda and Kirsten England whose work was highlighted in these slides.

And the entire Art team:

Whose ideas inspired many of these features, and who turned them into reality! Especially Jared Sobotta and Matthew Johns who contributed to these slides… and the tech!

Acknowledgements

The SIGGRAPH organizers:

Natalya Tatarchuk , Ginger Alford, Leona Caffey, Carlos Gonzalez-Ochoa, Kristy Pron, and Susan Reiser for all their patience and support!

Naughty Dog is Hiring!

http://www.naughtydog.com/careers

References

[Tatarchuk2006] – Practical Parallax Occlusion Mapping for Highly Detailed Surface Rendering

[VanDongen2008] – Interior Mapping: A new technique for rendering realistic buildings

[Jimenez2013] – Next Generation Character Rendering

[Brinck2016] – The Technical Art of Uncharted 4

[El-Garawany2016] – Deferred Lighting in Uncharted 4

[Gonzalez-Ochoa2016] – Rendering Rapids in Uncharted 4

[Kovalovs2020] – GPU Driven Effects of The Last of Us Part II

Contact Us!

waylon_brinck@naughtydog.com
steven_tang@naughtydog.com

And check out our team’s talks at SIGGRAPH 2020!

Artem Kovalovs: GPU Driven Effects of The Last of Us Part II
Artem Kovalovs: Volumetric Effects of The Last of Us Part II
Hawar Doghramachi: Lighting Technology of The Last of Us Part II
Parikshit Saraswat: Low-Level Optimizations in The Last of Us Part II